APPLICATION NOTE

AT09331: ASF USB Stack Manual

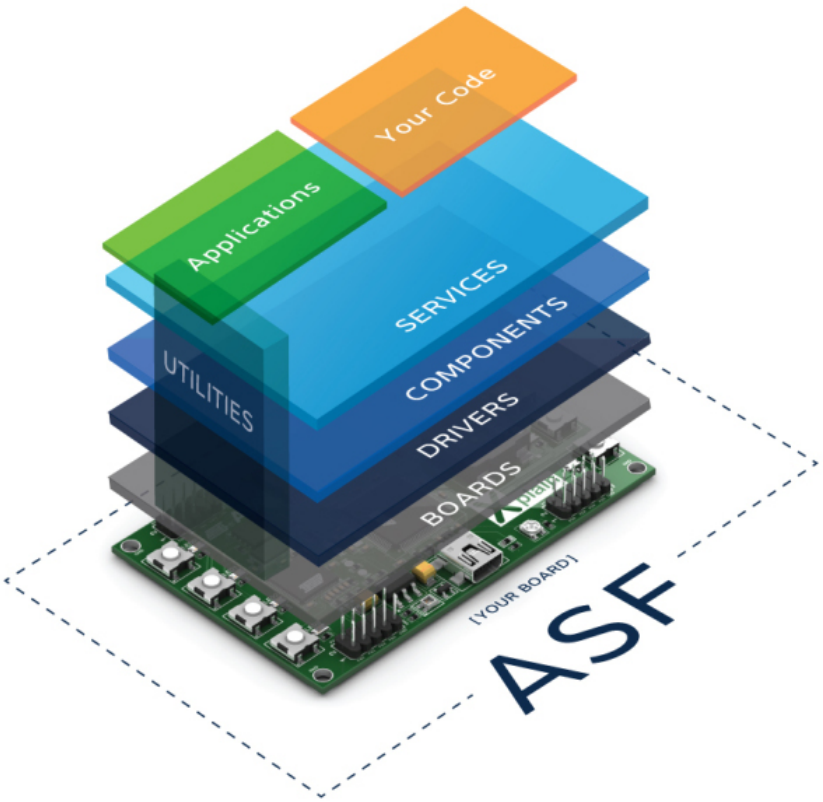
ASF PROGRAMMERS MANUAL

Preface

The Atmel® Software Framework (ASF) is a collection of free embedded software for Atmel microcontroller devices. It simplifies the usage of Atmel products, providing an abstraction to the hardware and high-value middleware.

ASF is designed to be used for evaluation, prototyping, design, and production phases. ASF is integrated in the Atmel Studio IDE with a graphical user interface or available as a standalone package for several commercial and open source compilers.

This document describes the API interfaces to the USB Stack for applications, included in ASF as middleware service.



For more information on ASF USB Stack and ASF, refer to the online documentation at following link:

- ASF-USB[1]
- ASF(www.atmel.com/asf)[2].

Table of Contents

Software License

1. USB Device Controller (UDC)

The UDC provides a high-level abstraction of the USB device. You can use these functions to control the main device state (start/attach/wakeup).

All USB Device Interface (UDI) in USB Device Stack is based on UDC to support USB enumeration.

This documentation describes common USB Device usage based on UDC, as follow:

- API Overview
- USB Device Basic Setup
- USB Device Advanced Use Cases

1.1 API Overview

1.1.1 Function Definitions

1.1.1.1 Function udc_attach()

Attach device to the bus when possible.

```
void udc_attach(void)
```

Warning If a VBUS control is included in driver, then it will attach device when an acceptable VBUS level from the host is detected.

1.1.1.2 Function udc_detach()

Detaches the device from the bus.

```
void udc_detach(void)
```

The driver must remove pull-up on USB line D- or D+.

1.1.1.3 Function udc_get_interface_desc()

Returns a pointer on the current interface descriptor.

```
usb_iface_desc_t UDC_DESC_STORAGE * udc_get_interface_desc(void)
```

Returns Pointer on the current interface descriptor.

1.1.1.4 Function udc_include_vbus_monitoring()

Authorizes the VBUS event.

```
bool udc_include_vbus_monitoring(void)
```

Returns True, if the VBUS monitoring is possible.

See VBUS Monitoring for more details.

1.1.1.5 Function udc_remotewakeup()

The USB driver sends a resume signal called "Upstream Resume".

```
void udc_remotewakeup(void)
```

This is authorized only when the remote wakeup feature is enabled by host.

1.1.1.6 Function udc_start()

Start the USB Device stack.

```
void udc_start(void)
```

1.1.1.7 Function udc_stop()

Stop the USB Device stack.

```
void udc_stop(void)
```

1.2 USB Device Basic Setup

1.2.1 Custom Configuration

The following USB Device configuration must be included in the conf_usb.h file of the application:

1. USB_DEVICE_VENDOR_ID (Word).

Vendor ID provided by USB org (Atmel 0x03EB).

2. USB_DEVICE_PRODUCT_ID (Word).

Product ID (Referenced in usb_atmel.h).

3. USB_DEVICE_MAJOR_VERSION (Byte).

Major version of the device.

4. USB_DEVICE_MINOR_VERSION (Byte).

Minor version of the device.

5. USB_DEVICE_MANUFACTURE_NAME (string).

ASCII name for the manufacture.

6. USB_DEVICE_PRODUCT_NAME (string).

ASCII name for the product.

7. USB_DEVICE_SERIAL_NAME (string).

ASCII name to enable and set a serial number.

8. USB_DEVICE_POWER (Numeric).

(unit mA) Maximum device power.

9. USB_DEVICE_ATTR (Byte).

USB attributes available:

- USB_CONFIG_ATTR_SELF_POWERED
- USB_CONFIG_ATTR_REMOTE_WAKEUP

Note If remote wake is enabled, this defines remotewakeup callbacks.

10. USB_DEVICE_LOW_SPEED (Only defined).

Force the USB Device to run in low speed.

11. USB_DEVICE_HS_SUPPORT (Only defined).

Authorize the USB Device to run in high speed.

12. USB_DEVICE_MAX_EP (Byte).

Define the maximum endpoint number used by the USB Device.

This one is already defined in the UDI default configuration. E.g.:

- When endpoint control 0x00, endpoint 0x01, and endpoint 0x82 is used, then USB_DEVICE_MAX_EP=2
- When only endpoint control 0x00 is used, then USB_DEVICE_MAX_EP=0
- When endpoint 0x01 and endpoint 0x81 is used, then USB_DEVICE_MAX_EP=1 (configuration not possible on USBB interface)

1.2.2 VBUS Monitoring

The VBUS monitoring is used only for USB SELF Power application.

- By default the USB device is automatically attached when VBUS is high or when USB starts for devices without internal VBUS monitoring. conf_usb.h file does not contain definition USB_DEVICE_ATTACH_AUTO_DISABLE.

```
//#define USB_DEVICE_ATTACH_AUTO_DISABLE
```

- Add custom VBUS monitoring. conf_usb.h file contains define USB_DEVICE_ATTACH_AUTO_DISABLE:

```
#define USB_DEVICE_ATTACH_AUTO_DISABLE
```

User C-file contains:

```
// Authorize VBUS monitoring
if (!udc_include_vbus_monitoring()) {
  // Implement custom VBUS monitoring via GPIO or other
}
Event_VBUS_present() // VBUS interrupt or GPIO interrupt or other
{
  // Attach USB Device
  udc_attach();
}
```

- Case of battery charging. conf_usb.h file contains define USB_DEVICE_ATTACH_AUTO_DISABLE:

```
#define USB_DEVICE_ATTACH_AUTO_DISABLE
```

User C-file contains:

```
Event VBUS present() // VBUS interrupt or GPIO interrupt or ..
{
  // Authorize battery charging, but wait key press to start USB.
}
Event Key press()
{
  // Stop batteries charging
  // Start USB
  udc_attach();
}
```

1.2.3 USB Device Basic Setup

1.2.3.1 USB Device Controller (UDC) - Prerequisites

Common prerequisites for all USB devices.

This module is based on USB device stack full interrupt driven, and supporting sleepmgr. For AVR® and Atmel® | SMART SAM3/4 devices the clock services is supported. For SAMD21 devices the clock driver is supported.

The following procedure must be executed to set up the project correctly:

- Specify the clock configuration:
  - XMEGA® USB devices need 48MHz clock input.

    XMEGA USB devices need CPU frequency higher than 12MHz.

    You can use either an internal RC 48MHz auto calibrated by Start of Frames or an external OSC.
  - UC3 and SAM3/4 devices without USB high speed support need 48MHz clock input.

    You must use a PLL and an external OSC.
  - UC3 and SAM3/4 devices with USB high speed support need 12MHz clock input.

    You must use an external OSC.
  - UC3 devices with USBC hardware need CPU frequency higher than 25MHz.
  - SAMD21 devices without USB high speed support need 48MHz clock input.

    You should use DFLL with USBCRM.
- In conf_board.h, the define CONF_BOARD_USB_PORT must be added to enable USB lines. (Not mandatory for all boards)
- Enable interrupts
- Initialize the clock service

The usage of sleepmgr service is optional, but recommended to reduce power consumption:

- Initialize the sleep manager service
- Activate sleep mode when the application is in IDLE state

conf_clock.h Examples.

For AVR and SAM3/4 devices, add to the initialization code:

```
sysclk_init();
irq_initialize_vectors();
cpu_irq_enable();
```

```
board_init();
sleepmgr_init(); // Optional
```

For SAMD21 devices, add to the initialization code:

```
system_init();
irq_initialize_vectors();
cpu_irq_enable();
sleepmgr_init(); // Optional
```

Add to the main IDLE loop:

```
sleepmgr_enter_sleep(); // Optional
```

1.2.3.2 USB Device Controller (UDC) - Example Code

Common example code for all USB devices.

Content of conf_usb.h:

```
#define USB_DEVICE_VENDOR_ID 0x03EB
#define USB_DEVICE_PRODUCT_ID 0xXXXX
#define USB_DEVICE_MAJOR_VERSION 1
#define USB_DEVICE_MINOR_VERSION 0
#define USB_DEVICE_POWER 100
#define USB_DEVICE_ATTR USB_CONFIG_ATTR_BUS_POWERED
```

Add to application C-file:

```
void usb_init(void)
{
  udc_start();
}
```

1.2.3.3 USB Device Controller (UDC) - Workflow

Common workflow for all USB devices.

1. Ensure that conf_usb.h is available and contains the following configuration, which is the main USB device configuration:

```
// Vendor ID provided by USB org (Atmel 0x03EB)
#define USB_DEVICE_VENDOR_ID 0x03EB // Type Word
// Product ID (Atmel PID referenced in usb_atmel.h)
#define USB_DEVICE_PRODUCT_ID 0xXXXX // Type Word
// Major version of the device
#define USB_DEVICE_MAJOR_VERSION 1 // Type Byte
// Minor version of the device
#define USB_DEVICE_MINOR_VERSION 0 // Type Byte
// Maximum device power (mA)
#define USB_DEVICE_POWER 100 // Type 9-bits
// USB attributes to enable features
#define USB_DEVICE_ATTR USB_CONFIG_ATTR_BUS_POWERED // Flags
```

2. Call the USB device stack start function to enable stack and start USB:

```
udc_start();
```

Note In case of USB dual roles (Device and Host) managed through USB OTG connector (USB ID pin), the call of udc_start() must be removed and replaced by uhc_start(). Refer to section "Dual roles" for further information in the application note: Atmel AVR4950: ASF - USB Host Stack[1]

1.2.4 conf_clock.h Examples

Content of XMEGA conf_clock.h:

```
// Configuration based on internal RC:
// USB clock need of 48MHz
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_RCOSC
#define CONFIG_OSC_RC32_CAL         48000000UL
#define CONFIG_OSC_AUTOCAL_RC32MHZ_REF_OSC  OSC_ID_USBSOF
// CPU clock need of clock > 12MHz to run with USB (Here 24MHz)
#define CONFIG_SYSCLK_SOURCE     SYSCLK_SRC_RC32MHZ
#define CONFIG_SYSCLK_PSADIV     SYSCLK_PSADIV_2
#define CONFIG_SYSCLK_PSBCDIV    SYSCLK_PSBCDIV_1_1
```

Content of conf_clock.h for AT32UC3A0, AT32UC3A1, and AT32UC3B devices (USBB):

```
// Configuration based on 12MHz external OSC:
#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
#define CONFIG_PLL1_MUL             8
#define CONFIG_PLL1_DIV             2
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV           1 // Fusb = Fsys/(2 ^ USB_div)
```

Content of conf_clock.h for AT32UC3A3 and AT32UC3A4 devices (USBB with high speed support):

```
// Configuration based on 12MHz external OSC:
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
#define CONFIG_USBCLK_DIV           1 // Fusb = Fsys/(2 ^ USB_div)
```

Content of conf_clock.h for AT32UC3C, ATUCXXD, ATUCXXL3U, and ATUCXXL4U devices (USBC):

```
// Configuration based on 12MHz external OSC:
#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
#define CONFIG_PLL1_MUL             8
#define CONFIG_PLL1_DIV             2
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV           1 // Fusb = Fsys/(2 ^ USB_div)
// CPU clock need of clock > 25MHz to run with USBC
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL1
```

Content of conf_clock.h for SAM3S, SAM3SD, and SAM4S devices (UPD: USB Peripheral Device):

```
// PLL1 (B) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
#define CONFIG_PLL1_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL1_MUL             16
#define CONFIG_PLL1_DIV             2
// USB Clock Source Options   (Fusb = FpllX / USB_div)
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV           2
```

[1] http://www.atmel.com/images/doc8486.pdf

Content of conf_clock.h for SAM3U device (UPDHS: USB Peripheral Device High Speed):

```
// USB Clock Source fixed at UPLL.
```

Content of conf_clock.h for SAM3X and SAM3A devices (UOTGHS: USB OTG High Speed):

```
// USB Clock Source fixed at UPLL.
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_UPLL
#define CONFIG_USBCLK_DIV           1
```

Content of conf_clocks.h for SAMD21 devices (USB):

```
// System clock bus configuration
#  define CONF_CLOCK_FLASH_WAIT_STATES            2

// USB Clock Source fixed at DFLL.
// SYSTEM_CLOCK_SOURCE_DFLL configuration - Digital Frequency Locked Loop
#  define CONF_CLOCK_DFLL_ENABLE                  true
#  define CONF_CLOCK_DFLL_LOOP_MODE               SYSTEM_CLOCK_DFLL_LOOP_MODE_USB_RECOVERY
#  define CONF_CLOCK_DFLL_ON_DEMAND               true

// Set this to true to configure the GCLK when running clocks_init.
// If set to false, none of the GCLK generators will be configured in clocks_init().
#  define CONF_CLOCK_CONFIGURE_GCLK               true

// Configure GCLK generator 0 (Main Clock)
#  define CONF_CLOCK_GCLK_0_ENABLE                true
#  define CONF_CLOCK_GCLK_0_RUN_IN_STANDBY        true
#  define CONF_CLOCK_GCLK_0_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_DFLL
#  define CONF_CLOCK_GCLK_0_PRESCALER             1
#  define CONF_CLOCK_GCLK_0_OUTPUT_ENABLE         false
```

1.3 USB Device Advanced Use Cases

- Change USB Speed
- Use USB Strings
- Use USB Remote Wakeup Feature
- Bus Power Application Recommendations
- USB Dynamic Serial Number

1.3.1 Change USB Speed

In this use case, the USB device is used with different USB speeds.

1.3.1.1 Setup Steps

Prior to implement this use case, be sure to have already applied the UDI module "basic use case".

1.3.1.2 Usage Steps

Example Code

Content of conf_usb.h:

```
#if // Low speed
```

```
#define USB_DEVICE_LOW_SPEED
// #define USB_DEVICE_HS_SUPPORT

#elif // Full speed
// #define USB_DEVICE_LOW_SPEED
// #define USB_DEVICE_HS_SUPPORT
#elif // High speed
// #define USB_DEVICE_LOW_SPEED
#define USB_DEVICE_HS_SUPPORT

#endif
```

Workflow

1. Ensure that conf_usb.h is available and contains the following parameters required for a USB device low speed (1.5Mbit/s):

```
#define USB_DEVICE_LOW_SPEED
//#define  USB_DEVICE_HS_SUPPORT
```

2. Ensure that conf_usb.h contains the following parameters required for a USB device full speed (12Mbit/s):

```
//#define USB_DEVICE_LOW_SPEED
//#define  USB_DEVICE_HS_SUPPORT
```

3. Ensure that conf_usb.h contains the following parameters required for a USB device high speed (480Mbit/s):

```
//#define USB_DEVICE_LOW_SPEED
#define  USB_DEVICE_HS_SUPPORT
```

1.3.2 Use USB Strings

In this use case, the usual USB strings are added in the USB device.

1.3.2.1 Setup Steps

Prior to implement this use case, be sure to have already applied the UDI module "basic use case".

1.3.2.2 Usage Steps

Example Code

Content of conf_usb.h:

```
#define  USB_DEVICE_MANUFACTURE_NAME      "Manufacture name"
#define  USB_DEVICE_PRODUCT_NAME          "Product name"
#define  USB_DEVICE_SERIAL_NAME           "12...EF"
```

Workflow

1. Ensure that conf_usb.h is available and contains the following parameters required to enable different USB strings:

```
// Static ASCII name for the manufacture
#define  USB_DEVICE_MANUFACTURE_NAME "Manufacture name"
```

```
// Static ASCII name for the product
```

```
#define  USB_DEVICE_PRODUCT_NAME "Product name"
```

```
// Static ASCII name to enable and set a serial number
#define  USB_DEVICE_SERIAL_NAME "12...EF"
```

1.3.3 Use USB Remote Wakeup Feature

In this use case, the USB remote wakeup feature is enabled.

1.3.3.1 Setup Steps

Prior to implement this use case, be sure to have already applied the UDI module "basic use case".

1.3.3.2 Usage Steps

Example Code

Content of conf_usb.h:

```
#define  USB_DEVICE_ATTR \
(USB_CONFIG_ATTR_REMOTE_WAKEUP | USB_CONFIG_ATTR_..._POWERED)
#define UDC_REMOTEWAKEUP_ENABLE() my_callback_remotewakeup_enable()
extern void my_callback_remotewakeup_enable(void);
#define UDC_REMOTEWAKEUP_DISABLE() my_callback_remotewakeup_disable()
extern void my_callback_remotewakeup_disable(void);
```

Add to application C-file:

```
void my_callback_remotewakeup_enable(void)
{
// Enable application wakeup events (e.g. enable GPIO interrupt)
}
void my_callback_remotewakeup_disable(void)
{
// Disable application wakeup events (e.g. disable GPIO interrupt)
}

void my_interrupt_event(void)
{
   udc_remotewakeup();
}
```

Workflow

1. Ensure that conf_usb.h is available and contains the following parameters required to enable the remote wakeup feature:

```
// Authorizes the remote wakeup feature
#define  USB_DEVICE_ATTR (USB_CONFIG_ATTR_REMOTE_WAKEUP | USB_CONFIG_ATTR_..._POWERED)
```

```
// Define callback called when the host enables the remotewakeup feature
#define UDC_REMOTEWAKEUP_ENABLE() my_callback_remotewakeup_enable()
extern void my_callback_remotewakeup_enable(void);
```

```
// Define callback called when the host disables the remotewakeup feature
#define UDC_REMOTEWAKEUP_DISABLE() my_callback_remotewakeup_disable()
```

```
extern void my_callback_remotewakeup_disable(void);
```

2. Send a remote wakeup (USB upstream):

```
udc_remotewakeup();
```

1.3.4 Bus Power Application Recommendations

In this use case, the USB device bus power feature is enabled. This feature requires a correct power consumption management.

1.3.4.1 Setup Steps

Prior to implement this use case, be sure to have already applied the UDI module "basic use case".

1.3.4.2 Usage Steps

Example Code

Content of conf_usb.h:

```
#define  USB_DEVICE_ATTR (USB_CONFIG_ATTR_BUS_POWERED)
#define  UDC_SUSPEND_EVENT()         user_callback_suspend_action()
extern void user_callback_suspend_action(void)
#define  UDC_RESUME_EVENT()          user_callback_resume_action()
extern void user_callback_resume_action(void)
```

Add to application C-file:

```
void user_callback_suspend_action(void)
{
   // Disable hardware component to reduce power consumption
}
void user_callback_resume_action(void)
{
   // Re-enable hardware component
}
```

Workflow

1. Ensure that conf_usb.h is available and contains the following parameters:

```
// Authorizes the BUS power feature
#define  USB_DEVICE_ATTR (USB_CONFIG_ATTR_BUS_POWERED)
```

```
// Define callback called when the host suspend the USB line
#define UDC_SUSPEND_EVENT() user_callback_suspend_action()
extern void user_callback_suspend_action(void);
```

```
// Define callback called when the host or device resume the USB line
#define UDC_RESUME_EVENT() user_callback_resume_action()
extern void user_callback_resume_action(void);
```

2. Reduce power consumption in suspend mode (max. 2.5mA on VBUS):

```
void user_callback_suspend_action(void)
```

```
{
   turn_off_components();
}
```

1.3.5 USB Dynamic Serial Number

In this use case, the USB serial strings are dynamic. For a static serial string refer to Use USB Strings.

1.3.5.1 Setup Steps

Prior to implement this use case, be sure to have already applied the UDI module "basic use case".

1.3.5.2 Usage Steps

Example Code

Content of conf_usb.h:

```
#define  USB_DEVICE_SERIAL_NAME
#define  USB_DEVICE_GET_SERIAL_NAME_POINTER serial_number
#define  USB_DEVICE_GET_SERIAL_NAME_LENGTH  12
extern uint8_t serial_number[];
```

Add to application C-file:

```
uint8_t serial_number[USB_DEVICE_GET_SERIAL_NAME_LENGTH];
void init_build_usb_serial_number(void)
{
serial_number[0] = 'A';
serial_number[1] = 'B';
...
serial_number[USB_DEVICE_GET_SERIAL_NAME_LENGTH-1] = 'C';
}
```

Workflow

1. Ensure that conf_usb.h is available and contains the following parameters required to enable a USB serial number string dynamically:

```
#define  USB_DEVICE_SERIAL_NAME // Define this empty
#define  USB_DEVICE_GET_SERIAL_NAME_POINTER serial_number // Give serial array pointer
#define  USB_DEVICE_GET_SERIAL_NAME_LENGTH  12 // Give size of serial array
extern uint8_t serial_number[]; // Declare external serial array
```

2. Before starting USB stack, initialize the serial array:

```
uint8_t serial_number[USB_DEVICE_GET_SERIAL_NAME_LENGTH];
void init_build_usb_serial_number(void)
{
serial_number[0] = 'A';
serial_number[1] = 'B';
...
serial_number[USB_DEVICE_GET_SERIAL_NAME_LENGTH-1] = 'C';
}
```

2. USB Device Interface (UDI) for Communication Class Device (CDC)

USB Device Interface (UDI) for Communication Class Device (CDC) provides an interface for the configuration and management of USB CDC serial device.

The outline of this documentation is as follows:

- API Overview
- Quick Start Guide for USB Device Communication Class Device Module (UDI CDC)
- Configuration File Examples

For more details for Atmel® Software Framework (ASF) USB Device Stack and USB Device CDC, refer to following application notes:

- AVR4900: ASF - USB Device Stack[1]
- AVR4907: ASF - USB Device CDC Application[2]
- AVR4920: ASF - USB Device Stack - Compliance and Performance Figures[3]
- AVR4921: ASF - USB Device Stack Differences between ASF V1 and V2[4]

2.1 API Overview

2.1.1 Structure Definitions

2.1.1.1 Struct udi_cdc_comm_desc_t

Interface descriptor with associated functional and endpoint descriptors for the CDC Communication Class interface.

Table 2-1. Members

Type	Name	Description
usb_cdc_acm_desc_t	acm	CDC Abstract Control Model functional descriptor
usb_cdc_call_mgmt_desc_t	call_mgmt	CDC Call Management functional descriptor
usb_ep_desc_t	ep_notify	Notification endpoint descriptor
usb_cdc_hdr_desc_t	header	CDC Header functional descriptor
usb_iface_desc_t	iface	Standard interface descriptor
usb_cdc_union_desc_t	union_desc	CDC Union functional descriptor

2.1.1.2 Struct udi_cdc_data_desc_t

Interface descriptor with associated endpoint descriptors for the CDC Data Class interface.

Table 2-2. Members

Type	Name	Description
usb_ep_desc_t	ep_in	Data IN endpoint descriptors
usb_ep_desc_t	ep_out	Data OUT endpoint descriptors

[1] http://www.atmel.com/dyn/resources/prod_documents/doc8360.pdf
[2] http://www.atmel.com/dyn/resources/prod_documents/doc8447.pdf
[3] http://www.atmel.com/dyn/resources/prod_documents/doc8410.pdf
[4] http://www.atmel.com/dyn/resources/prod_documents/doc8411.pdf

Type	Name	Description
usb_iface_desc_t	iface	Standard interface descriptor

2.1.2 Macro Definitions

2.1.2.1 Content of Interface Descriptors

Up to seven CDC interfaces can be implemented on a USB device.

Macro UDI_CDC_IAD_STRING_ID_0

```
#define UDI_CDC_IAD_STRING_ID_0 0
```

No string associated to IAD interface.

Macro UDI_CDC_COMM_STRING_ID_0

```
#define UDI_CDC_COMM_STRING_ID_0 0
```

No string associated to COMM interface.

Macro UDI_CDC_DATA_STRING_ID_0

```
#define UDI_CDC_DATA_STRING_ID_0 0
```

No string associated to DATA interface.

Macro UDI_CDC_IAD_DESC_0

```
#define UDI_CDC_IAD_DESC_0 UDI_CDC_IAD_DESC(0)
```

IAD descriptor for port 0.

Macro UDI_CDC_COMM_DESC_0

```
#define UDI_CDC_COMM_DESC_0 UDI_CDC_COMM_DESC(0)
```

COMM descriptors for port 0.

Macro UDI_CDC_DATA_DESC_0_FS

```
#define UDI_CDC_DATA_DESC_0_FS UDI_CDC_DATA_DESC_FS(0)
```

DATA descriptor for port 0 of a full speed device.

Macro UDI_CDC_DATA_DESC_0_HS

```
#define UDI_CDC_DATA_DESC_0_HS UDI_CDC_DATA_DESC_HS(0)
```

DATA descriptor for port 0 of a high speed device.

Macro UDI_CDC_IAD_STRING_ID_1

```
#define UDI_CDC_IAD_STRING_ID_1 0
```

No string associated to IAD interface.

Macro UDI_CDC_COMM_STRING_ID_1

```
#define UDI_CDC_COMM_STRING_ID_1 0
```

No string associated to COMM interface.

Macro UDI_CDC_DATA_STRING_ID_1

```
#define UDI_CDC_DATA_STRING_ID_1 0
```

Macro UDI_CDC_IAD_DESC_1

```
#define UDI_CDC_IAD_DESC_1 UDI_CDC_IAD_DESC(1)
```

Macro UDI_CDC_COMM_DESC_1

```
#define UDI_CDC_COMM_DESC_1 UDI_CDC_COMM_DESC(1)
```

Macro UDI_CDC_DATA_DESC_1_FS

```
#define UDI_CDC_DATA_DESC_1_FS UDI_CDC_DATA_DESC_FS(1)
```

Macro UDI_CDC_DATA_DESC_1_HS

```
#define UDI_CDC_DATA_DESC_1_HS UDI_CDC_DATA_DESC_HS(1)
```

Macro UDI_CDC_IAD_STRING_ID_2

```
#define UDI_CDC_IAD_STRING_ID_2 0
```

No string associated to IAD interface.

Macro UDI_CDC_COMM_STRING_ID_2

```
#define UDI_CDC_COMM_STRING_ID_2 0
```

No string associated to COMM interface.

Macro UDI_CDC_DATA_STRING_ID_2

```
#define UDI_CDC_DATA_STRING_ID_2 0
```

Macro UDI_CDC_IAD_DESC_2

```
#define UDI_CDC_IAD_DESC_2 UDI_CDC_IAD_DESC(2)
```

Macro UDI_CDC_COMM_DESC_2

```
#define UDI_CDC_COMM_DESC_2 UDI_CDC_COMM_DESC(2)
```

Macro UDI_CDC_DATA_DESC_2_FS

```
#define UDI_CDC_DATA_DESC_2_FS UDI_CDC_DATA_DESC_FS(2)
```

Macro UDI_CDC_DATA_DESC_2_HS

```
#define UDI_CDC_DATA_DESC_2_HS UDI_CDC_DATA_DESC_HS(2)
```

Macro UDI_CDC_IAD_STRING_ID_3

```
#define UDI_CDC_IAD_STRING_ID_3 0
```

No string associated to IAD interface.

Macro UDI_CDC_COMM_STRING_ID_3

```
#define UDI_CDC_COMM_STRING_ID_3 0
```

No string associated to COMM interface.

Macro UDI_CDC_DATA_STRING_ID_3

```
#define UDI_CDC_DATA_STRING_ID_3 0
```

Macro UDI_CDC_IAD_DESC_3

```
#define UDI_CDC_IAD_DESC_3 UDI_CDC_IAD_DESC(3)
```

Macro UDI_CDC_COMM_DESC_3

```
#define UDI_CDC_COMM_DESC_3 UDI_CDC_COMM_DESC(3)
```

Macro UDI_CDC_DATA_DESC_3_FS

```
#define UDI_CDC_DATA_DESC_3_FS UDI_CDC_DATA_DESC_FS(3)
```

Macro UDI_CDC_DATA_DESC_3_HS

```
#define UDI_CDC_DATA_DESC_3_HS UDI_CDC_DATA_DESC_HS(3)
```

Macro UDI_CDC_IAD_STRING_ID_4

```
#define UDI_CDC_IAD_STRING_ID_4 0
```

No string associated to IAD interface.

Macro UDI_CDC_COMM_STRING_ID_4

```
#define UDI_CDC_COMM_STRING_ID_4 0
```

No string associated to COMM interface.

Macro UDI_CDC_DATA_STRING_ID_4

```
#define UDI_CDC_DATA_STRING_ID_4 0
```

Macro UDI_CDC_IAD_DESC_4

```
#define UDI_CDC_IAD_DESC_4 UDI_CDC_IAD_DESC(4)
```

Macro UDI_CDC_COMM_DESC_4

```
#define UDI_CDC_COMM_DESC_4 UDI_CDC_COMM_DESC(4)
```

Macro UDI_CDC_DATA_DESC_4_FS

```
#define UDI_CDC_DATA_DESC_4_FS UDI_CDC_DATA_DESC_FS(4)
```

Macro UDI_CDC_DATA_DESC_4_HS

```
#define UDI_CDC_DATA_DESC_4_HS UDI_CDC_DATA_DESC_HS(4)
```

Macro UDI_CDC_IAD_STRING_ID_5

```
#define UDI_CDC_IAD_STRING_ID_5 0
```

No string associated to IAD interface.

Macro UDI_CDC_COMM_STRING_ID_5

```
#define UDI_CDC_COMM_STRING_ID_5 0
```

No string associated to COMM interface.

Macro UDI_CDC_DATA_STRING_ID_5

```
#define UDI_CDC_DATA_STRING_ID_5 0
```

Macro UDI_CDC_IAD_DESC_5

```
#define UDI_CDC_IAD_DESC_5 UDI_CDC_IAD_DESC(5)
```

Macro UDI_CDC_COMM_DESC_5

```
#define UDI_CDC_COMM_DESC_5 UDI_CDC_COMM_DESC(5)
```

Macro UDI_CDC_DATA_DESC_5_FS

```
#define UDI_CDC_DATA_DESC_5_FS UDI_CDC_DATA_DESC_FS(5)
```

Macro UDI_CDC_DATA_DESC_5_HS

```
#define UDI_CDC_DATA_DESC_5_HS UDI_CDC_DATA_DESC_HS(5)
```

Macro UDI_CDC_IAD_STRING_ID_6

```
#define UDI_CDC_IAD_STRING_ID_6 0
```

No string associated to IAD interface.

Macro UDI_CDC_COMM_STRING_ID_6

```
#define UDI_CDC_COMM_STRING_ID_6 0
```

Macro UDI_CDC_DATA_STRING_ID_6

```
#define UDI_CDC_DATA_STRING_ID_6 0
```

Macro UDI_CDC_IAD_DESC_6

```
#define UDI_CDC_IAD_DESC_6 UDI_CDC_IAD_DESC(6)
```

Macro UDI_CDC_COMM_DESC_6

```
#define UDI_CDC_COMM_DESC_6 UDI_CDC_COMM_DESC(6)
```

Macro UDI_CDC_DATA_DESC_6_FS

```
#define UDI_CDC_DATA_DESC_6_FS UDI_CDC_DATA_DESC_FS(6)
```

Macro UDI_CDC_DATA_DESC_6_HS

```
#define UDI_CDC_DATA_DESC_6_HS UDI_CDC_DATA_DESC_HS(6)
```

2.1.2.2 Macro UDI_CDC_COMM_DESC

```
#define UDI_CDC_COMM_DESC(port) \
  { \
   .iface.bLength                = sizeof(usb_iface_desc_t),\
   .iface.bDescriptorType        = USB_DT_INTERFACE,\
   .iface.bAlternateSetting      = 0,\
   .iface.bNumEndpoints          = 1,\
   .iface.bInterfaceClass        = CDC_CLASS_COMM,\
   .iface.bInterfaceSubClass     = CDC_SUBCLASS_ACM,\
   .iface.bInterfaceProtocol     = CDC_PROTOCOL_V25TER,\
   .header.bFunctionLength       = sizeof(usb_cdc_hdr_desc_t),\
   .header.bDescriptorType       = CDC_CS_INTERFACE,\
   .header.bDescriptorSubtype    = CDC_SCS_HEADER,\
   .header.bcdCDC                = LE16(0x0110),\
   .call_mgmt.bFunctionLength    = sizeof(usb_cdc_call_mgmt_desc_t),\
   .call_mgmt.bDescriptorType    = CDC_CS_INTERFACE,\
   .call_mgmt.bDescriptorSubtype = CDC_SCS_CALL_MGMT,\
   .call_mgmt.bmCapabilities     = \
   CDC_CALL_MGMT_SUPPORTED | CDC_CALL_MGMT_OVER_DCI,\
   .acm.bFunctionLength          = sizeof(usb_cdc_acm_desc_t),\
   .acm.bDescriptorType          = CDC_CS_INTERFACE,\
   .acm.bDescriptorSubtype       = CDC_SCS_ACM,\
   .acm.bmCapabilities           = CDC_ACM_SUPPORT_LINE_REQUESTS,\
   .union_desc.bFunctionLength   = sizeof(usb_cdc_union_desc_t),\
   .union_desc.bDescriptorType   = CDC_CS_INTERFACE,\
   .union_desc.bDescriptorSubtype= CDC_SCS_UNION,\
   .ep_notify.bLength            = sizeof(usb_ep_desc_t),\
   .ep_notify.bDescriptorType    = USB_DT_ENDPOINT,\
   .ep_notify.bmAttributes       = USB_EP_TYPE_INTERRUPT,\
   .ep_notify.wMaxPacketSize     = LE16(UDI_CDC_COMM_EP_SIZE),\
   .ep_notify.bInterval          = 0x10,\
   .ep_notify.bEndpointAddress   = UDI_CDC_COMM_EP_##port,\
   .iface.bInterfaceNumber       = UDI_CDC_COMM_IFACE_NUMBER_##port,\
   .call_mgmt.bDataInterface     = UDI_CDC_DATA_IFACE_NUMBER_##port,\
   .union_desc.bMasterInterface  = UDI_CDC_COMM_IFACE_NUMBER_##port,\
   .union_desc.bSlaveInterface0  = UDI_CDC_DATA_IFACE_NUMBER_##port,\
   .iface.iInterface             = UDI_CDC_COMM_STRING_ID_##port,\
   }
```

Content of CDC COMM interface descriptor for all speeds.

2.1.2.3 Macro UDI_CDC_COMM_EP_SIZE

```
#define UDI_CDC_COMM_EP_SIZE 64
```

CDC communication endpoints size for all speeds.

2.1.2.4 Macro UDI_CDC_DATA_DESC_COMMON

```
#define UDI_CDC_DATA_DESC_COMMON \
  .iface.bLength                = sizeof(usb_iface_desc_t),\
   .iface.bDescriptorType        = USB_DT_INTERFACE,\
   .iface.bAlternateSetting      = 0,\
   .iface.bNumEndpoints          = 2,\
   .iface.bInterfaceClass        = CDC_CLASS_DATA,\
   .iface.bInterfaceSubClass     = 0,\
   .iface.bInterfaceProtocol     = 0,\
   .ep_in.bLength                = sizeof(usb_ep_desc_t),\
   .ep_in.bDescriptorType        = USB_DT_ENDPOINT,\
   .ep_in.bmAttributes           = USB_EP_TYPE_BULK,\
   .ep_in.bInterval              = 0,\
   .ep_out.bLength               = sizeof(usb_ep_desc_t),\
   .ep_out.bDescriptorType       = USB_DT_ENDPOINT,\
   .ep_out.bmAttributes          = USB_EP_TYPE_BULK,\
   .ep_out.bInterval             = 0,
```

Content of CDC DATA interface descriptors.

2.1.2.5 Macro UDI_CDC_DATA_DESC_FS

```
#define UDI_CDC_DATA_DESC_FS(port) \
  { \
   UDI_CDC_DATA_DESC_COMMON \
   .ep_in.wMaxPacketSize         = LE16(UDI_CDC_DATA_EPS_FS_SIZE),\
   .ep_out.wMaxPacketSize        = LE16(UDI_CDC_DATA_EPS_FS_SIZE),\
   .ep_in.bEndpointAddress       = UDI_CDC_DATA_EP_IN_##port,\
   .ep_out.bEndpointAddress      = UDI_CDC_DATA_EP_OUT_##port,\
   .iface.bInterfaceNumber       = UDI_CDC_DATA_IFACE_NUMBER_##port,\
   .iface.iInterface             = UDI_CDC_DATA_STRING_ID_##port,\
   }
```

Content of CDC DATA interface descriptors for FS.

2.1.2.6 Macro UDI_CDC_DATA_DESC_HS

```
#define UDI_CDC_DATA_DESC_HS(port) \
  { \
   UDI_CDC_DATA_DESC_COMMON \
   .ep_in.wMaxPacketSize         = LE16(UDI_CDC_DATA_EPS_HS_SIZE),\
   .ep_out.wMaxPacketSize        = LE16(UDI_CDC_DATA_EPS_HS_SIZE),\
   .ep_in.bEndpointAddress       = UDI_CDC_DATA_EP_IN_##port,\
   .ep_out.bEndpointAddress      = UDI_CDC_DATA_EP_OUT_##port,\
```

```
  .iface.bInterfaceNumber       = UDI_CDC_DATA_IFACE_NUMBER_##port,\
  .iface.iInterface             = UDI_CDC_DATA_STRING_ID_##port,\
  }
```

Content of CDC DATA interface descriptors for HS.

2.1.2.7 Macro UDI_CDC_DATA_EPS_FS_SIZE

```
#define UDI_CDC_DATA_EPS_FS_SIZE 64
```

CDC data endpoints size for FS speed (8B, 16B, 32B, 64B).

2.1.2.8 Macro UDI_CDC_DATA_EPS_HS_SIZE

```
#define UDI_CDC_DATA_EPS_HS_SIZE 512
```

CDC data endpoints size for HS speed (512B only).

2.1.2.9 Macro UDI_CDC_IAD_DESC

```
#define UDI_CDC_IAD_DESC(port) \
  { \
   .bLength                     = sizeof(usb_iad_desc_t),\
   .bDescriptorType             = USB_DT_IAD,\
   .bInterfaceCount             = 2,\
   .bFunctionClass              = CDC_CLASS_COMM,\
   .bFunctionSubClass           = CDC_SUBCLASS_ACM,\
   .bFunctionProtocol           = CDC_PROTOCOL_V25TER,\
   .bFirstInterface             = UDI_CDC_COMM_IFACE_NUMBER_##port,\
   .iFunction                   = UDI_CDC_IAD_STRING_ID_##port,\
   }
```

Content of CDC IAD interface descriptor for all speeds.

2.1.3 Function Definitions

2.1.3.1 Interface for Application with Single CDC Interface Support

Function udi_cdc_ctrl_signal_dcd()

Notify a state change of DCD signal.

```
void udi_cdc_ctrl_signal_dcd(
  bool b_set)
```

Table 2-3. Parameters

Data direction	Parameter name	Description
[in]	b_set	DCD is enabled if true, else disabled

Function udi_cdc_ctrl_signal_dsr()

Notify a state change of DSR signal.

```
void udi_cdc_ctrl_signal_dsr(
  bool b_set)
```

Table 2-4. Parameters

Data direction	Parameter name	Description
[in]	b_set	DSR is enabled if true, else disabled

Function udi_cdc_signal_framing_error()

Notify a framing error.

```
void udi_cdc_signal_framing_error(void)
```

Function udi_cdc_signal_parity_error()

Notify a parity error.

```
void udi_cdc_signal_parity_error(void)
```

Function udi_cdc_signal_overrun()

Notify a overrun.

```
void udi_cdc_signal_overrun(void)
```

Function udi_cdc_get_nb_received_data()

Gets the number of byte received.

```
iram_size_t udi_cdc_get_nb_received_data(void)
```

Returns

The number of data available.

Function udi_cdc_is_rx_ready()

This function checks if a character has been received on the CDC line.

```
bool udi_cdc_is_rx_ready(void)
```

Returns 1 if a byte is ready to be read.

Function udi_cdc_getc()

Waits and gets a value on CDC line.

```
int udi_cdc_getc(void)
```

Returns Value read on CDC line.

Function udi_cdc_read_buf()

Reads a RAM buffer on CDC line.

```
iram_size_t udi_cdc_read_buf(
  void * buf,
  iram_size_t size)
```

Table 2-5. Parameters

Data direction	Parameter name	Description
[out]	buf	Values read
[in]	size	Number of value read

Returns The number of data remaining.

Function udi_cdc_get_free_tx_buffer()

Gets the number of free byte in TX buffer.

```
iram_size_t udi_cdc_get_free_tx_buffer(void)
```

Returns The number of free byte in TX buffer.

Function udi_cdc_is_tx_ready()

This function checks if a new character sent is possible. The type int is used to support scanf redirection from compiler LIB.

```
bool udi_cdc_is_tx_ready(void)
```

Returns 1 if a new character can be sent.

Function udi_cdc_putc()

Puts a byte on CDC line.

```
int udi_cdc_putc(
  int value)
```

The type int is used to support printf redirection from compiler LIB.

Table 2-6. Parameters

Data direction	Parameter name	Description
[in]	value	Value to put

Returns 1 if function was successfully done, otherwise 0.

Function udi_cdc_write_buf()

Writes a RAM buffer on CDC line.

```
iram_size_t udi_cdc_write_buf(
  const void * buf,
  iram_size_t size)
```

Table 2-7. Parameters

Data direction	Parameter name	Description
[in]	buf	Values to write
[in]	size	Number of value to write

Returns The number of data remaining.

2.1.3.2 Interface for Application with Multi CDC Interfaces Support

Function udi_cdc_multi_ctrl_signal_dcd()

Notify a state change of DCD signal.

```
void udi_cdc_multi_ctrl_signal_dcd(
  uint8_t port,
```

```
  bool b_set)
```

Table 2-8. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number to manage
[in]	b_set	DCD is enabled if true, else disabled

Function udi_cdc_multi_ctrl_signal_dsr()

Notify a state change of DSR signal.

```
void udi_cdc_multi_ctrl_signal_dsr(
  uint8_t port,
  bool b_set)
```

Table 2-9. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number to manage
[in]	b_set	DSR is enabled if true, else disabled

Function udi_cdc_multi_signal_framing_error()

Notify a framing error.

```
void udi_cdc_multi_signal_framing_error(
  uint8_t port)
```

Table 2-10. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number to manage

Function udi_cdc_multi_signal_parity_error()

Notify a parity error.

```
void udi_cdc_multi_signal_parity_error(
  uint8_t port)
```

Table 2-11. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number to manage

Function udi_cdc_multi_signal_overrun()

Notify a overrun.

```
void udi_cdc_multi_signal_overrun(
  uint8_t port)
```

Table 2-12. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number to manage

Function udi_cdc_multi_get_nb_received_data()

Gets the number of byte received.

```
iram_size_t udi_cdc_multi_get_nb_received_data(
  uint8_t port)
```

Table 2-13. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number to manage

Returns The number of data available.

Function udi_cdc_multi_is_rx_ready()

This function checks if a character has been received on the CDC line.

```
bool udi_cdc_multi_is_rx_ready(
  uint8_t port)
```

Table 2-14. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number to manage

Returns 1 if a byte is ready to be read.

Function udi_cdc_multi_getc()

Waits and gets a value on CDC line.

```
int udi_cdc_multi_getc(
  uint8_t port)
```

Table 2-15. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number to manage

Returns Value read on CDC line.

Function udi_cdc_multi_read_buf()

Reads a RAM buffer on CDC line.

```
iram_size_t udi_cdc_multi_read_buf(
  uint8_t port,
  void * buf,
  iram_size_t size)
```

Table 2-16. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number to manage
[out]	buf	Values read
[in]	size	Number of values read

Returns The number of data remaining.

Function udi_cdc_multi_get_free_tx_buffer()

Gets the number of free byte in TX buffer.

```
iram_size_t udi_cdc_multi_get_free_tx_buffer(
  uint8_t port)
```

Table 2-17. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number to manage

Returns The number of free byte in TX buffer.

Function udi_cdc_multi_is_tx_ready()

This function checks if a new character sent is possible.

```
bool udi_cdc_multi_is_tx_ready(
  uint8_t port)
```

The type int is used to support scanf redirection from compiler LIB.

Table 2-18. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number to manage

Returns

1 if a new character can be sent.

Function udi_cdc_multi_putc()

Puts a byte on CDC line, and the type int is used to support printf redirection from compiler LIB.

```
int udi_cdc_multi_putc(
  uint8_t port,
  int value)
```

Table 2-19. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number to manage
[in]	value	Value to put

Returns

1 if function was successfully done, otherwise 0.

Function udi_cdc_multi_write_buf()

Writes a RAM buffer on CDC line.

```
iram_size_t udi_cdc_multi_write_buf(
  uint8_t port,
  const void * buf,
  iram_size_t size)
```

Table 2-20. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number to manage
[in]	buf	Values to write

Data direction	Parameter name	Description
[in]	size	Number of value to write

Returns The number of data remaining.

2.2 Quick Start Guide for USB Device Communication Class Device Module (UDI CDC)

This is the quick start guide for the USB Device Interface CDC Module (UDI CDC) with step-by-step instructions on how to configure and use the modules in a selection of use cases.

The use cases contain or highlights several code fragments. The code fragments in the steps for setup can be copied into a custom initialization function, while the steps for usage can be copied into, e.g., the main application function.

2.2.1 Basic Use Case

In this basic use case, the "USB CDC (Single Interface Device)" module is used with only one communication port. The "USB CDC (Composite Device)" module usage is described in Advanced Use Cases.

2.2.1.1 Setup Steps

As a USB device, it follows common USB device setup steps. Refer to USB Device Basic Setup.

2.2.1.2 Usage Steps

Example Code

Content of conf_usb.h:

```
#define UDI_CDC_ENABLE_EXT(port) my_callback_cdc_enable()
extern bool my_callback_cdc_enable(void);
#define UDI_CDC_DISABLE_EXT(port) my_callback_cdc_disable()
extern void my_callback_cdc_disable(void);
#define  UDI_CDC_LOW_RATE

#define  UDI_CDC_DEFAULT_RATE             115200
#define  UDI_CDC_DEFAULT_STOPBITS         CDC_STOP_BITS_1
#define  UDI_CDC_DEFAULT_PARITY           CDC_PAR_NONE
#define  UDI_CDC_DEFAULT_DATABITS         8

#include "udi_cdc_conf.h" // At the end of conf_usb.h file
```

Add to application C-file:

```
static bool my_flag_autorize_cdc_transfert = false;
bool my_callback_cdc_enable(void)
{
  my_flag_autorize_cdc_transfert = true;
  return true;
}

void my_callback_cdc_disable(void)
{
  my_flag_autorize_cdc_transfert = false;
}

void task(void)
```

```
{
  if (my_flag_autorize_cdc_transfert) {
  udi_cdc_putc('A');
  udi_cdc_getc();
 }
}
```

Workflow

1. Ensure that conf_usb.h is available and contains the following configuration which is the USB device CDC configuration:

```
#define USB_DEVICE_SERIAL_NAME  "12...EF" // Disk SN for CDC
```

Note The USB serial number is mandatory when a CDC interface is used.

```
#define UDI_CDC_ENABLE_EXT(port) my_callback_cdc_enable()
extern bool my_callback_cdc_enable(void);
```

Note After the device enumeration (detecting and identifying USB devices), the USB host starts the device configuration. When the USB CDC interface from the device is accepted by the host, the USB host enables this interface and the UDI_CDC_ENABLE_EXT() callback function is called and return true. Thus, when this event is received, the data transfer on CDC interface are authorized.

```
#define UDI_CDC_DISABLE_EXT(port) my_callback_cdc_disable()
extern void my_callback_cdc_disable(void);
```

Note When the USB device is unplugged or is reset by the USB host, the USB interface is disabled and the UDI_CDC_DISABLE_EXT() callback function is called. Thus, the data transfer must be stopped on CDC interface.

```
#define  UDI_CDC_LOW_RATE
```

Note Define it when the transfer CDC Device to Host is a low rate (<512000 bauds) to reduce CDC buffers size.

```
#define  UDI_CDC_DEFAULT_RATE             115200
#define  UDI_CDC_DEFAULT_STOPBITS         CDC_STOP_BITS_1
#define  UDI_CDC_DEFAULT_PARITY           CDC_PAR_NONE
#define  UDI_CDC_DEFAULT_DATABITS         8
```

Note Default configuration of communication port at startup.

2. Send or wait data on CDC line:

```
// Waits and gets a value on CDC line
int udi_cdc_getc(void);
// Reads a RAM buffer on CDC line
iram_size_t udi_cdc_read_buf(int* buf, iram_size_t size);
// Puts a byte on CDC line
int udi_cdc_putc(int value);
// Writes a RAM buffer on CDC line
iram_size_t udi_cdc_write_buf(const int* buf, iram_size_t size);
```

2.2.2 Advanced Use Cases

For multiple interface use of UDI CDC module, see the following:

- CDC in a Composite Device

For more advanced use of the UDI CDC module, see the following:

- USB Device Advanced Use Cases

2.2.3 CDC in a Composite Device

A USB Composite Device is a USB Device which uses more than one USB class. In this use case, the "USB CDC (Composite Device)" module is used to create a USB composite device. Thus, this USB module can be associated with another "Composite Device" module, like "USB HID Mouse (Composite Device)".

Also, you can refer to application note AVR4902 ASF - USB Composite Device[5].

2.2.3.1 Setup Steps

For the setup code of this use case to work, the Basic Use Case must be followed.

2.2.3.2 Usage Steps

Example Code

Content of conf_usb.h:

```
#define USB_DEVICE_EP_CTRL_SIZE  64
#define USB_DEVICE_NB_INTERFACE (X+2)
#define USB_DEVICE_MAX_EP (X+3)

#define  UDI_CDC_DATA_EP_IN_0          (1 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_0         (2 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_0             (3 | USB_EP_DIR_IN)  // Notify endpoint
#define  UDI_CDC_COMM_IFACE_NUMBER_0   X+0
#define  UDI_CDC_DATA_IFACE_NUMBER_0   X+1

#define UDI_COMPOSITE_DESC_T \
   usb_iad_desc_t udi_cdc_iad; \
   udi_cdc_comm_desc_t udi_cdc_comm; \
   udi_cdc_data_desc_t udi_cdc_data; \
   ...
#define UDI_COMPOSITE_DESC_FS \
   .udi_cdc_iad               = UDI_CDC_IAD_DESC_0, \
   .udi_cdc_comm              = UDI_CDC_COMM_DESC_0, \
   .udi_cdc_data              = UDI_CDC_DATA_DESC_0_FS, \
   ...
#define UDI_COMPOSITE_DESC_HS \
   .udi_cdc_iad               = UDI_CDC_IAD_DESC_0, \
```

[5] http://www.atmel.com/dyn/resources/prod_documents/doc8445.pdf

```
   .udi_cdc_comm              = UDI_CDC_COMM_DESC_0, \
   .udi_cdc_data              = UDI_CDC_DATA_DESC_0_HS, \
   ...
#define UDI_COMPOSITE_API \
   &udi_api_cdc_comm,       \
   &udi_api_cdc_data,       \
   ...
```

Workflow

1. Ensure that conf_usb.h is available and contains the following parameters required for a USB composite device configuration:

```
// Endpoint control size, This must be:
// - 8, 16, 32 or 64 for full speed device (8 is recommended to save RAM)
// - 64 for a high speed device
#define USB_DEVICE_EP_CTRL_SIZE  64
// Total Number of interfaces on this USB device.
// Add 2 for CDC.
#define USB_DEVICE_NB_INTERFACE (X+2)
// Total number of endpoints on this USB device.
// This must include each endpoint for each interface.
// Add 3 for CDC.
#define USB_DEVICE_MAX_EP (X+3)
```

2. Ensure that conf_usb.h contains the description of composite device:

```
// The endpoint numbers chosen by you for the CDC.
// The endpoint numbers starting from 1.
#define  UDI_CDC_DATA_EP_IN_0            (1 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_0           (2 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_0               (3 | USB_EP_DIR_IN)  // Notify endpoint
// The interface index of an interface starting from 0
#define  UDI_CDC_COMM_IFACE_NUMBER_0     X+0
#define  UDI_CDC_DATA_IFACE_NUMBER_0     X+1
```

3. Ensure that conf_usb.h contains the following parameters required for a USB composite device configuration:

```
// USB Interfaces descriptor structure
#define UDI_COMPOSITE_DESC_T \
  ...
  usb_iad_desc_t udi_cdc_iad; \
  udi_cdc_comm_desc_t udi_cdc_comm; \
  udi_cdc_data_desc_t udi_cdc_data; \
  ...
// USB Interfaces descriptor value for Full Speed
#define UDI_COMPOSITE_DESC_FS \
   ...
   .udi_cdc_iad               = UDI_CDC_IAD_DESC_0, \
   .udi_cdc_comm              = UDI_CDC_COMM_DESC_0, \
   .udi_cdc_data              = UDI_CDC_DATA_DESC_0_FS, \
   ...
// USB Interfaces descriptor value for High Speed
#define UDI_COMPOSITE_DESC_HS \
   ...
   .udi_cdc_iad               = UDI_CDC_IAD_DESC_0, \
   .udi_cdc_comm              = UDI_CDC_COMM_DESC_0, \
   .udi_cdc_data              = UDI_CDC_DATA_DESC_0_HS, \
```

```
   ...
// USB Interface APIs
#define UDI_COMPOSITE_API \
   ...
   &udi_api_cdc_comm,       \
   &udi_api_cdc_data,       \
   ...
```

Note The descriptors order given in the four lists above must be the same as the order defined by all interface indexes. The interface index orders are defined through UDI_X_IFACE_NUMBER defines. Also, the CDC requires a USB Interface Association Descriptor (IAD) for composite device.

2.3 Configuration File Examples

2.3.1 conf_usb.h

2.3.1.1 UDI CDC Single

```
#ifndef _CONF_USB_H_
#define _CONF_USB_H_

#include "compiler.h"

#warning You must refill the following definitions with a correct values

#define  USB_DEVICE_VENDOR_ID             USB_VID_ATMEL
#define  USB_DEVICE_PRODUCT_ID            USB_PID_ATMEL_ASF_HIDGENERIC
#define  USB_DEVICE_MAJOR_VERSION         1
#define  USB_DEVICE_MINOR_VERSION         0
#define  USB_DEVICE_POWER                 100 // Consumption on Vbus line (mA)
#define  USB_DEVICE_ATTR                  \
    (USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_BUS_POWERED)
//  (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_SELF_POWERED)
//  (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_BUS_POWERED)

// #define  USB_DEVICE_MANUFACTURE_NAME      "Manufacture name"
// #define  USB_DEVICE_PRODUCT_NAME          "Product name"
// #define  USB_DEVICE_SERIAL_NAME           "12...EF"

//#define  USB_DEVICE_LOW_SPEED

#if (UC3A3||UC3A4)
//#define  USB_DEVICE_HS_SUPPORT
#endif


// #define  UDC_VBUS_EVENT(b_vbus_high)      user_callback_vbus_action(b_vbus_high)
// extern void user_callback_vbus_action(bool b_vbus_high);
// #define  UDC_SOF_EVENT()                  user_callback_sof_action()
// extern void user_callback_sof_action(void);
// #define  UDC_SUSPEND_EVENT()              user_callback_suspend_action()
// extern void user_callback_suspend_action(void);
// #define  UDC_RESUME_EVENT()               user_callback_resume_action()
// extern void user_callback_resume_action(void);
// #define  UDC_REMOTEWAKEUP_ENABLE()        user_callback_remotewakeup_enable()
// extern void user_callback_remotewakeup_enable(void);
```

```
// #define  UDC_REMOTEWAKEUP_DISABLE()       user_callback_remotewakeup_disable()
// extern void user_callback_remotewakeup_disable(void);
// #define  UDC_GET_EXTRA_STRING()



#define  UDI_HID_GENERIC_ENABLE_EXT()       true
#define  UDI_HID_GENERIC_DISABLE_EXT()
#define  UDI_HID_GENERIC_REPORT_OUT(ptr)
#define  UDI_HID_GENERIC_SET_FEATURE(f)
/*
 * #define UDI_HID_GENERIC_ENABLE_EXT() my_callback_generic_enable()
 * extern bool my_callback_generic_enable(void);
 * #define UDI_HID_GENERIC_DISABLE_EXT() my_callback_generic_disable()
 * extern void my_callback_generic_disable(void);
 * #define  UDI_HID_GENERIC_REPORT_OUT(ptr) my_callback_generic_report_out(ptr)
 * extern void my_callback_generic_report_out(uint8_t *report);
 * #define  UDI_HID_GENERIC_SET_FEATURE(f) my_callback_generic_set_feature(f)
 * extern void my_callback_generic_set_feature(uint8_t *report_feature);
 */

#define  UDI_HID_REPORT_IN_SIZE             64
#define  UDI_HID_REPORT_OUT_SIZE            64
#define  UDI_HID_REPORT_FEATURE_SIZE        4

#define  UDI_HID_GENERIC_EP_SIZE            64




#include "udi_hid_generic_conf.h"

#endif // _CONF_USB_H_
```

2.3.1.2 UDI CDC Multiple (Composite)

```
#ifndef _CONF_USB_H_
#define _CONF_USB_H_

#include "compiler.h"

#warning You must refill the following definitions with a correct values

#define  USB_DEVICE_VENDOR_ID             USB_VID_ATMEL
#define  USB_DEVICE_PRODUCT_ID            0xFFFF
#define  USB_DEVICE_MAJOR_VERSION         1
#define  USB_DEVICE_MINOR_VERSION         0
#define  USB_DEVICE_POWER                 100 // Consumption on VBUS line (mA)
#define  USB_DEVICE_ATTR                  \
        (USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_BUS_POWERED)
// (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_BUS_POWERED)

// #define  USB_DEVICE_MANUFACTURE_NAME      "Manufacture name"
// #define  USB_DEVICE_PRODUCT_NAME          "Product name"
// #define  USB_DEVICE_SERIAL_NAME           "12...EF" // Disk SN for MSC

//#define  USB_DEVICE_LOW_SPEED
```

```
#if (UC3A3||UC3A4)
//#define  USB_DEVICE_HS_SUPPORT
#endif


// #define  UDC_VBUS_EVENT(b_vbus_high)      user_callback_vbus_action(b_vbus_high)
// extern void user_callback_vbus_action(bool b_vbus_high);
// #define  UDC_SOF_EVENT()                  user_callback_sof_action()
// extern void user_callback_sof_action(void);
// #define  UDC_SUSPEND_EVENT()              user_callback_suspend_action()
// extern void user_callback_suspend_action(void);
// #define  UDC_RESUME_EVENT()               user_callback_resume_action()
// extern void user_callback_resume_action(void);
// #define  UDC_REMOTEWAKEUP_ENABLE()        user_callback_remotewakeup_enable()
// extern void user_callback_remotewakeup_enable(void);
// #define  UDC_REMOTEWAKEUP_DISABLE()       user_callback_remotewakeup_disable()
// extern void user_callback_remotewakeup_disable(void);
// #define  UDC_GET_EXTRA_STRING()

#define  USB_DEVICE_EP_CTRL_SIZE       64

#define  USB_DEVICE_NB_INTERFACE       1 // 1 or more

#define  USB_DEVICE_MAX_EP             1 // 0 to max endpoint requested by interfaces




#define  UDI_CDC_PORT_NB 1

#define  UDI_CDC_ENABLE_EXT(port)             true
#define  UDI_CDC_DISABLE_EXT(port)
#define  UDI_CDC_RX_NOTIFY(port)
#define  UDI_CDC_TX_EMPTY_NOTIFY(port)
#define  UDI_CDC_SET_CODING_EXT(port,cfg)
#define  UDI_CDC_SET_DTR_EXT(port,set)
#define  UDI_CDC_SET_RTS_EXT(port,set)
/*
 * #define UDI_CDC_ENABLE_EXT(port) my_callback_cdc_enable()
 * extern bool my_callback_cdc_enable(void);
 * #define UDI_CDC_DISABLE_EXT(port) my_callback_cdc_disable()
 * extern void my_callback_cdc_disable(void);
 * #define  UDI_CDC_RX_NOTIFY(port) my_callback_rx_notify(port)
 * extern void my_callback_rx_notify(uint8_t port);
 * #define  UDI_CDC_TX_EMPTY_NOTIFY(port) my_callback_tx_empty_notify(port)
 * extern void my_callback_tx_empty_notify(uint8_t port);
 * #define  UDI_CDC_SET_CODING_EXT(port,cfg) my_callback_config(port,cfg)
 * extern void my_callback_config(uint8_t port, usb_cdc_line_coding_t * cfg);
 * #define  UDI_CDC_SET_DTR_EXT(port,set) my_callback_cdc_set_dtr(port,set)
 * extern void my_callback_cdc_set_dtr(uint8_t port, bool b_enable);
 * #define  UDI_CDC_SET_RTS_EXT(port,set) my_callback_cdc_set_rts(port,set)
 * extern void my_callback_cdc_set_rts(uint8_t port, bool b_enable);
 */

#define  UDI_CDC_LOW_RATE

#define  UDI_CDC_DEFAULT_RATE             115200
#define  UDI_CDC_DEFAULT_STOPBITS         CDC_STOP_BITS_1
#define  UDI_CDC_DEFAULT_PARITY           CDC_PAR_NONE
#define  UDI_CDC_DEFAULT_DATABITS         8
```

```
#define  UDI_CDC_DATA_EP_IN_0         (1 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_0        (2 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_0            (3 | USB_EP_DIR_IN)  // Notify endpoint
#define  UDI_CDC_DATA_EP_IN_2         (4 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_2        (5 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_2            (6 | USB_EP_DIR_IN)  // Notify endpoint
#define  UDI_CDC_DATA_EP_IN_3         (7 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_3        (8 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_3            (9 | USB_EP_DIR_IN)  // Notify endpoint

#define  UDI_CDC_COMM_IFACE_NUMBER_0  0
#define  UDI_CDC_DATA_IFACE_NUMBER_0  1
#define  UDI_CDC_COMM_IFACE_NUMBER_2  2
#define  UDI_CDC_DATA_IFACE_NUMBER_2  3
#define  UDI_CDC_COMM_IFACE_NUMBER_3  4
#define  UDI_CDC_DATA_IFACE_NUMBER_3  5




#define UDI_MSC_GLOBAL_VENDOR_ID           \
   'A', 'T', 'M', 'E', 'L', ' ', ' ', ' '
#define UDI_MSC_GLOBAL_PRODUCT_VERSION           \
   '1', '.', '0', '0'

#define  UDI_MSC_ENABLE_EXT()         true
#define  UDI_MSC_DISABLE_EXT()
#define  UDI_MSC_NOTIFY_TRANS_EXT()
/*
 * #define UDI_MSC_ENABLE_EXT() my_callback_msc_enable()
 * extern bool my_callback_msc_enable(void);
 * #define UDI_MSC_DISABLE_EXT() my_callback_msc_disable()
 * extern void my_callback_msc_disable(void);
 * #define  UDI_MSC_NOTIFY_TRANS_EXT()    msc_notify_trans()
 * extern void msc_notify_trans(void) {
 */

#define  UDI_MSC_EP_IN                (1 | USB_EP_DIR_IN)
#define  UDI_MSC_EP_OUT               (2 | USB_EP_DIR_OUT)

#define  UDI_MSC_IFACE_NUMBER         0




#define  UDI_HID_MOUSE_ENABLE_EXT()       true
#define  UDI_HID_MOUSE_DISABLE_EXT()
// #define UDI_HID_MOUSE_ENABLE_EXT() my_callback_mouse_enable()
// extern bool my_callback_mouse_enable(void);
// #define UDI_HID_MOUSE_DISABLE_EXT() my_callback_mouse_disable()
// extern void my_callback_mouse_disable(void);

#define  UDI_HID_MOUSE_EP_IN          (1 | USB_EP_DIR_IN)

#define  UDI_HID_MOUSE_IFACE_NUMBER    0




#define  UDI_HID_KBD_ENABLE_EXT()       true
```

```
#define  UDI_HID_KBD_DISABLE_EXT()
// #define UDI_HID_KBD_ENABLE_EXT() my_callback_keyboard_enable()
// extern bool my_callback_keyboard_enable(void);
// #define UDI_HID_KBD_DISABLE_EXT() my_callback_keyboard_disable()
// extern void my_callback_keyboard_disable(void);
#define  UDI_HID_KBD_CHANGE_LED(value)
// #define  UDI_HID_KBD_CHANGE_LED(value) my_callback_keyboard_led(value)
// extern void my_callback_keyboard_led(uint8_t value)

#define  UDI_HID_KBD_EP_IN           (1 | USB_EP_DIR_IN)

#define  UDI_HID_KBD_IFACE_NUMBER    0



#define  UDI_HID_GENERIC_ENABLE_EXT()       true
#define  UDI_HID_GENERIC_DISABLE_EXT()
#define  UDI_HID_GENERIC_REPORT_OUT(ptr)
#define  UDI_HID_GENERIC_SET_FEATURE(f)
/*
 * #define UDI_HID_GENERIC_ENABLE_EXT() my_callback_generic_enable()
 * extern bool my_callback_generic_enable(void);
 * #define UDI_HID_GENERIC_DISABLE_EXT() my_callback_generic_disable()
 * extern void my_callback_generic_disable(void);
 * #define  UDI_HID_GENERIC_REPORT_OUT(ptr) my_callback_generic_report_out(ptr)
 * extern void my_callback_generic_report_out(uint8_t *report);
 * #define  UDI_HID_GENERIC_SET_FEATURE(f) my_callback_generic_set_feature(f)
 * extern void my_callback_generic_set_feature(uint8_t *report_feature);
 */
#define  UDI_HID_REPORT_IN_SIZE             64
#define  UDI_HID_REPORT_OUT_SIZE            64
#define  UDI_HID_REPORT_FEATURE_SIZE        4
#define  UDI_HID_GENERIC_EP_SIZE            64

#define  UDI_HID_GENERIC_EP_OUT   (2 | USB_EP_DIR_OUT)
#define  UDI_HID_GENERIC_EP_IN    (1 | USB_EP_DIR_IN)

#define  UDI_HID_GENERIC_IFACE_NUMBER    0



#define  UDI_PHDC_ENABLE_EXT()          true
#define  UDI_PHDC_DISABLE_EXT()

#define  UDI_PHDC_DATAMSG_FORMAT       USB_PHDC_DATAMSG_FORMAT_11073_20601
#define  UDI_PHDC_SPECIALIZATION       {0x2345} // Define in 11073_20601

#define  UDI_PHDC_QOS_OUT              \
        (USB_PHDC_QOS_MEDIUM_BETTER|USB_PHDC_QOS_HIGH_BEST)
#define  UDI_PHDC_QOS_IN               \
        (USB_PHDC_QOS_LOW_GOOD|USB_PHDC_QOS_MEDIUM_BETTER|USB_PHDC_QOS_MEDIUM_BEST)

#define  UDI_PHDC_METADATA_DESC_BULK_IN   {0x01,0x02,0x03}
#define  UDI_PHDC_METADATA_DESC_BULK_OUT  {0x01,0x02,0x03}
#define  UDI_PHDC_METADATA_DESC_INT_IN    {0x01,0x02,0x03}

#define  UDI_PHDC_EP_BULK_OUT          (1 | USB_EP_DIR_OUT)
#define  UDI_PHDC_EP_BULK_IN           (2 | USB_EP_DIR_IN)
#if ((UDI_PHDC_QOS_IN&USB_PHDC_QOS_LOW_GOOD)==USB_PHDC_QOS_LOW_GOOD)
```

```
// Only if UDI_PHDC_QOS_IN include USB_PHDC_QOS_LOW_GOOD
#  define  UDI_PHDC_EP_INTERRUPT_IN      (3 | USB_EP_DIR_IN)
#endif

#define  UDI_PHDC_EP_SIZE_BULK_OUT     32
#define  UDI_PHDC_EP_SIZE_BULK_IN      32
#define  UDI_PHDC_EP_SIZE_INT_IN       8


#define  UDI_PHDC_IFACE_NUMBER          0



#define  UDI_VENDOR_ENABLE_EXT()          true
#define  UDI_VENDOR_DISABLE_EXT()
#define  UDI_VENDOR_SETUP_OUT_RECEIVED()  false
#define  UDI_VENDOR_SETUP_IN_RECEIVED()   false
/*
 * #define UDI_VENDOR_ENABLE_EXT() my_callback_vendor_enable()
 * extern bool my_callback_vendor_enable(void);
 * #define UDI_VENDOR_DISABLE_EXT() my_callback_vendor_disable()
 * extern void my_callback_vendor_disable(void);
 *
 * #define  UDI_VENDOR_SETUP_OUT_RECEIVED()  my_vendor_setup_out_received()
 * extern bool my_vendor_setup_out_received(void);
 * #define  UDI_VENDOR_SETUP_IN_RECEIVED()   my_vendor_setup_in_received()
 * extern bool my_vendor_setup_in_received(void);
 */

#define  UDI_VENDOR_EPS_SIZE_INT_FS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_FS   64
#define  UDI_VENDOR_EPS_SIZE_ISO_FS   256

#define  UDI_VENDOR_EPS_SIZE_INT_HS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_HS  512
#define  UDI_VENDOR_EPS_SIZE_ISO_HS    64

#define  UDI_VENDOR_EP_INTERRUPT_IN  (1 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_INTERRUPT_OUT (2 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_BULK_IN       (3 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_BULK_OUT      (4 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_ISO_IN        (5 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_ISO_OUT       (6 | USB_EP_DIR_OUT)


#define  UDI_VENDOR_IFACE_NUMBER     0



//... Eventually add other Interface Configuration



#define UDI_COMPOSITE_DESC_T

#define UDI_COMPOSITE_DESC_FS

#define UDI_COMPOSITE_DESC_HS

#define UDI_COMPOSITE_API
```

```
/* Example for device with cdc, msc and hid mouse interface
#define UDI_COMPOSITE_DESC_T \
    usb_iad_desc_t udi_cdc_iad; \
    udi_cdc_comm_desc_t udi_cdc_comm; \
    udi_cdc_data_desc_t udi_cdc_data; \
    udi_msc_desc_t udi_msc; \
    udi_hid_mouse_desc_t udi_hid_mouse

#define UDI_COMPOSITE_DESC_FS \
    .udi_cdc_iad               = UDI_CDC_IAD_DESC_0, \
    .udi_cdc_comm              = UDI_CDC_COMM_DESC_0, \
    .udi_cdc_data              = UDI_CDC_DATA_DESC_0_FS, \
    .udi_msc                   = UDI_MSC_DESC_FS, \
    .udi_hid_mouse             = UDI_HID_MOUSE_DESC

#define UDI_COMPOSITE_DESC_HS \
    .udi_cdc_iad               = UDI_CDC_IAD_DESC_0, \
    .udi_cdc_comm              = UDI_CDC_COMM_DESC_0, \
    .udi_cdc_data              = UDI_CDC_DATA_DESC_0_HS, \
    .udi_msc                   = UDI_MSC_DESC_HS, \
    .udi_hid_mouse             = UDI_HID_MOUSE_DESC

#define UDI_COMPOSITE_API  \
    &udi_api_cdc_comm,       \
    &udi_api_cdc_data,       \
    &udi_api_msc,            \
    &udi_api_hid_mouse
*/




/* Example of include for interface
#include "udi_msc.h"
#include "udi_hid_kbd.h"
#include "udi_hid_mouse.h"
#include "udi_cdc.h"
#include "udi_phdc.h"
#include "udi_vendor.h"
*/
/* Declaration of callbacks used by USB
#include "callback_def.h"
*/

#endif // _CONF_USB_H_
```

2.3.2 conf_clock.h

2.3.2.1 XMEGA (USB)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

#define CONFIG_USBCLK_SOURCE     USBCLK_SRC_RCOSC
#define CONFIG_OSC_RC32_CAL      48000000UL

#define CONFIG_OSC_AUTOCAL_RC32MHZ_REF_OSC  OSC_ID_USBSOF

#define CONFIG_SYSCLK_SOURCE     SYSCLK_SRC_RC32MHZ
```

```
#define CONFIG_SYSCLK_PSADIV     SYSCLK_PSADIV_2
#define CONFIG_SYSCLK_PSBCDIV    SYSCLK_PSBCDIV_1_2

/*

#define CONFIG_PLL0_SOURCE       PLL_SRC_XOSC
#define CONFIG_PLL0_MUL          6
#define CONFIG_PLL0_DIV          1

#define CONFIG_USBCLK_SOURCE     USBCLK_SRC_PLL

#define CONFIG_SYSCLK_SOURCE     SYSCLK_SRC_PLL
#define CONFIG_SYSCLK_PSADIV     SYSCLK_PSADIV_2
#define CONFIG_SYSCLK_PSBCDIV    SYSCLK_PSBCDIV_1_2
*/


#endif /* CONF_CLOCK_H_INCLUDED */
```

2.3.2.2 AT32UC3A0, AT32UC3A1, AT32UC3B Devices (USBB)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL0

// ===== PLL0 Options
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_MUL             4 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL0_DIV             1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
#define CONFIG_PLL1_SOURCE            PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
#define CONFIG_PLL1_MUL               8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL1_DIV               2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
//#define CONFIG_SYSCLK_CPU_DIV         0 /* Fcpu = Fsys/(2 ^ CPU_div) */
//#define CONFIG_SYSCLK_PBA_DIV         0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV         0 /* Fpbb = Fsys/(2 ^ PBB_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

2.3.2.3 AT32UC3A3, AT32UC3A4 Devices (USBB with High Speed Support)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL0

// ===== PLL0 Options
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_MUL             11 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL0_DIV             2  /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL1_MUL             8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL1_DIV             2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
#define CONFIG_SYSCLK_CPU_DIV         0 /* Fcpu = Fsys/(2 ^ CPU_div) */
#define CONFIG_SYSCLK_PBA_DIV         0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV       0 /* Fpbb = Fsys/(2 ^ PBB_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

2.3.2.4 AT32UC3C, ATUCXXD, ATUCXXL3U, ATUCXXL4U Devices (USBC)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC1
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_PLL0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL1
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RC8M

// ===== PLL0 Options
#define CONFIG_PLL0_SOURCE            PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_SOURCE          PLL_SRC_RC8M
#define CONFIG_PLL0_MUL               3 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL0_DIV               1 /* Fpll = (Fclk * PLL_mul) / PLL_div */
```

```
// ===== PLL1 Options
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL1_SOURCE          PLL_SRC_RC8M
//#define CONFIG_PLL1_MUL             3 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL1_DIV             1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
//#define CONFIG_SYSCLK_CPU_DIV       0 /* Fcpu = Fsys/(2 ^ CPU_div) */
//#define CONFIG_SYSCLK_PBA_DIV       0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV       0 /* Fpbb = Fsys/(2 ^ PBB_div) */
//#define CONFIG_SYSCLK_PBC_DIV       0 /* Fpbc = Fsys/(2 ^ PBC_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC1
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

2.3.2.5 SAM3S, SAM3SD, SAM4S Devices (UPD: USB Peripheral Device)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock (MCK) Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_BYPASS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_4M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_8M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_12M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_BYPASS
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLLACK
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLLBCK

// ===== System Clock (MCK) Prescaler Options   (Fmck = Fsys / (SYSCLK_PRES))
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_1
#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_2
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_4
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_8
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_16
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_32
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_64
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_3

// ===== PLL0 (A) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
// Use mul and div effective values here.
#define CONFIG_PLL0_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL0_MUL             32
```

```
#define CONFIG_PLL0_DIV             3

// ===== PLL1 (B) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
// Use mul and div effective values here.
#define CONFIG_PLL1_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL1_MUL             16
#define CONFIG_PLL1_DIV             2

// ===== USB Clock Source Options   (Fusb = FpllX / USB_div)
// Use div effective value here.
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV           2

// ===== Target frequency (System clock)
// - XTAL frequency: 12MHz
// - System clock source: PLLA
// - System clock prescaler: 2 (divided by 2)
// - PLLA source: XTAL
// - PLLA output: XTAL * 32 / 3
// - System clock is: 12 * 32 / 3 / 2 = 64MHz
// ===== Target frequency (USB Clock)
// - USB clock source: PLLB
// - USB clock divider: 2 (divided by 2)
// - PLLB output: XTAL * 16 / 2
// - USB clock: 12 * 16 / 2 / 2 = 48MHz


#endif /* CONF_CLOCK_H_INCLUDED */
```

2.3.2.6 SAM3U Device (UPDHS: USB Peripheral Device High Speed)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock (MCK) Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_BYPASS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_4M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_8M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_12M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_BYPASS
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLLACK
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_UPLLCK

// ===== System Clock (MCK) Prescaler Options   (Fmck = Fsys / (SYSCLK_PRES))
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_1
#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_2
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_4
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_8
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_16
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_32
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_64
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_3

// ===== PLL0 (A) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
// Use mul and div effective values here.
#define CONFIG_PLL0_SOURCE          PLL_SRC_MAINCK_XTAL
```

```
#define CONFIG_PLL0_MUL             16
#define CONFIG_PLL0_DIV             1

// ===== UPLL (UTMI) Hardware fixed at 480MHz.

// ===== USB Clock Source fixed at UPLL.

// ===== Target frequency (System clock)
// - XTAL frequency: 12MHz
// - System clock source: PLLA
// - System clock prescaler: 2 (divided by 2)
// - PLLA source: XTAL
// - PLLA output: XTAL * 16 / 1
// - System clock is: 12 * 16 / 1 / 2 = 96MHz
// ===== Target frequency (USB Clock)
// - USB clock source: UPLL
// - UPLL frequency: 480MHz
// - USB clock: 480MHz


#endif /* CONF_CLOCK_H_INCLUDED */
```

2.3.2.7 SAM3X, SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock (MCK) Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_BYPASS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_4M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_8M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_12M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_BYPASS
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLLACK
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_UPLLCK

// ===== System Clock (MCK) Prescaler Options   (Fmck = Fsys / (SYSCLK_PRES))
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_1
#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_2
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_4
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_8
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_16
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_32
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_64
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_3

// ===== PLL0 (A) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
// Use mul and div effective values here.
#define CONFIG_PLL0_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL0_MUL             14
#define CONFIG_PLL0_DIV             1

// ===== UPLL (UTMI) Hardware fixed at 480MHz.

// ===== USB Clock Source Options   (Fusb = FpllX / USB_div)
// Use div effective value here.
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
```

```
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_UPLL
#define CONFIG_USBCLK_DIV           1

// ===== Target frequency (System clock)
// - XTAL frequency: 12MHz
// - System clock source: PLLA
// - System clock prescaler: 2 (divided by 2)
// - PLLA source: XTAL
// - PLLA output: XTAL * 14 / 1
// - System clock is: 12 * 14 / 1 /2 = 84MHz
// ===== Target frequency (USB Clock)
// - USB clock source: UPLL
// - USB clock divider: 1 (not divided)
// - UPLL frequency: 480MHz
// - USB clock: 480 / 1 = 480MHz


#endif /* CONF_CLOCK_H_INCLUDED */
```

2.3.3 conf_clocks.h

2.3.3.1 SAMD21 Device (USB)

```
#include <clock.h>

#ifndef CONF_CLOCKS_H_INCLUDED
#  define CONF_CLOCKS_H_INCLUDED

/* System clock bus configuration */
#  define CONF_CLOCK_CPU_CLOCK_FAILURE_DETECT     false
#  define CONF_CLOCK_FLASH_WAIT_STATES            2
#  define CONF_CLOCK_CPU_DIVIDER                  SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBA_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBB_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1

/* SYSTEM_CLOCK_SOURCE_OSC8M configuration - Internal 8MHz oscillator */
#  define CONF_CLOCK_OSC8M_PRESCALER              SYSTEM_OSC8M_DIV_1
#  define CONF_CLOCK_OSC8M_ON_DEMAND              true
#  define CONF_CLOCK_OSC8M_RUN_IN_STANDBY         false

/* SYSTEM_CLOCK_SOURCE_XOSC configuration - External clock/oscillator */
#  define CONF_CLOCK_XOSC_ENABLE                  false
#  define CONF_CLOCK_XOSC_EXTERNAL_CRYSTAL        SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC_EXTERNAL_FREQUENCY      12000000UL
#  define CONF_CLOCK_XOSC_STARTUP_TIME            SYSTEM_XOSC_STARTUP_32768
#  define CONF_CLOCK_XOSC_AUTO_GAIN_CONTROL       true
#  define CONF_CLOCK_XOSC_ON_DEMAND               true
#  define CONF_CLOCK_XOSC_RUN_IN_STANDBY          false

/* SYSTEM_CLOCK_SOURCE_XOSC32K configuration - External 32KHz crystal/clock oscillator */
#  define CONF_CLOCK_XOSC32K_ENABLE               false
#  define CONF_CLOCK_XOSC32K_EXTERNAL_CRYSTAL     SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC32K_STARTUP_TIME         SYSTEM_XOSC32K_STARTUP_65536
#  define CONF_CLOCK_XOSC32K_AUTO_AMPLITUDE_CONTROL  false
#  define CONF_CLOCK_XOSC32K_ENABLE_1KHZ_OUPUT    false
#  define CONF_CLOCK_XOSC32K_ENABLE_32KHZ_OUTPUT  true
#  define CONF_CLOCK_XOSC32K_ON_DEMAND            true
#  define CONF_CLOCK_XOSC32K_RUN_IN_STANDBY       false

/* SYSTEM_CLOCK_SOURCE_OSC32K configuration - Internal 32KHz oscillator */
```

```
#  define CONF_CLOCK_OSC32K_ENABLE                false
#  define CONF_CLOCK_OSC32K_STARTUP_TIME          SYSTEM_OSC32K_STARTUP_130
#  define CONF_CLOCK_OSC32K_ENABLE_1KHZ_OUTPUT    true
#  define CONF_CLOCK_OSC32K_ENABLE_32KHZ_OUTPUT   true
#  define CONF_CLOCK_OSC32K_ON_DEMAND             true
#  define CONF_CLOCK_OSC32K_RUN_IN_STANDBY        false

/* SYSTEM_CLOCK_SOURCE_DFLL configuration - Digital Frequency Locked Loop */
#  define CONF_CLOCK_DFLL_ENABLE                  true
#  define CONF_CLOCK_DFLL_LOOP_MODE               SYSTEM_CLOCK_DFLL_LOOP_MODE_USB_RECOVERY
#  define CONF_CLOCK_DFLL_ON_DEMAND               true

/* DFLL open loop mode configuration */
#  define CONF_CLOCK_DFLL_COARSE_VALUE            (0x1f / 4)
#  define CONF_CLOCK_DFLL_FINE_VALUE              (0xff / 4)

/* DFLL closed loop mode configuration */
#  define CONF_CLOCK_DFLL_SOURCE_GCLK_GENERATOR   GCLK_GENERATOR_1
#  define CONF_CLOCK_DFLL_MULTIPLY_FACTOR         (48000000 / 32768)
#  define CONF_CLOCK_DFLL_QUICK_LOCK              true
#  define CONF_CLOCK_DFLL_TRACK_AFTER_FINE_LOCK   true
#  define CONF_CLOCK_DFLL_KEEP_LOCK_ON_WAKEUP     true
#  define CONF_CLOCK_DFLL_ENABLE_CHILL_CYCLE      true
#  define CONF_CLOCK_DFLL_MAX_COARSE_STEP_SIZE    (0x1f / 4)
#  define CONF_CLOCK_DFLL_MAX_FINE_STEP_SIZE      (0xff / 4)

/* SYSTEM_CLOCK_SOURCE_DPLL configuration - Digital Phase-Locked Loop */
#  define CONF_CLOCK_DPLL_ENABLE                  false
#  define CONF_CLOCK_DPLL_ON_DEMAND               true
#  define CONF_CLOCK_DPLL_RUN_IN_STANDBY          false
#  define CONF_CLOCK_DPLL_LOCK_BYPASS             false
#  define CONF_CLOCK_DPLL_WAKE_UP_FAST            false
#  define CONF_CLOCK_DPLL_LOW_POWER_ENABLE        false

#  define CONF_CLOCK_DPLL_LOCK_TIME               SYSTEM_CLOCK_SOURCE_DPLL_LOCK_TIME_NO_TIMEOUT
#  define CONF_CLOCK_DPLL_REFERENCE_CLOCK         SYSTEM_CLOCK_SOURCE_DPLL_REFERENCE_CLOCK_REF0
#  define CONF_CLOCK_DPLL_FILTER                  SYSTEM_CLOCK_SOURCE_DPLL_FILTER_DEFAULT

#  define CONF_CLOCK_DPLL_REFERENCE_FREQUENCY     32768
#  define CONF_CLOCK_DPLL_REFEREMCE_DIVIDER       1
#  define CONF_CLOCK_DPLL_OUTPUT_FREQUENCY        48000000

/* Set this to true to configure the GCLK when running clocks_init. If set to
 * false, none of the GCLK generators will be configured in clocks_init(). */
#  define CONF_CLOCK_CONFIGURE_GCLK               true

/* Configure GCLK generator 0 (Main Clock) */
#  define CONF_CLOCK_GCLK_0_ENABLE                true
#  define CONF_CLOCK_GCLK_0_RUN_IN_STANDBY        true
#  define CONF_CLOCK_GCLK_0_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_DFLL
#  define CONF_CLOCK_GCLK_0_PRESCALER             1
#  define CONF_CLOCK_GCLK_0_OUTPUT_ENABLE         false

/* Configure GCLK generator 1 */
#  define CONF_CLOCK_GCLK_1_ENABLE                false
#  define CONF_CLOCK_GCLK_1_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_1_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_XOSC32K
#  define CONF_CLOCK_GCLK_1_PRESCALER             1
#  define CONF_CLOCK_GCLK_1_OUTPUT_ENABLE         false

/* Configure GCLK generator 2 (RTC) */
```

```
#  define CONF_CLOCK_GCLK_2_ENABLE                false
#  define CONF_CLOCK_GCLK_2_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_2_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC32K
#  define CONF_CLOCK_GCLK_2_PRESCALER             32
#  define CONF_CLOCK_GCLK_2_OUTPUT_ENABLE         false

/* Configure GCLK generator 3 */
#  define CONF_CLOCK_GCLK_3_ENABLE                false
#  define CONF_CLOCK_GCLK_3_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_3_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_3_PRESCALER             1
#  define CONF_CLOCK_GCLK_3_OUTPUT_ENABLE         false

/* Configure GCLK generator 4 */
#  define CONF_CLOCK_GCLK_4_ENABLE                false
#  define CONF_CLOCK_GCLK_4_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_4_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_4_PRESCALER             1
#  define CONF_CLOCK_GCLK_4_OUTPUT_ENABLE         false

/* Configure GCLK generator 5 */
#  define CONF_CLOCK_GCLK_5_ENABLE                false
#  define CONF_CLOCK_GCLK_5_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_5_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_5_PRESCALER             1
#  define CONF_CLOCK_GCLK_5_OUTPUT_ENABLE         false

/* Configure GCLK generator 6 */
#  define CONF_CLOCK_GCLK_6_ENABLE                false
#  define CONF_CLOCK_GCLK_6_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_6_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_6_PRESCALER             1
#  define CONF_CLOCK_GCLK_6_OUTPUT_ENABLE         false

/* Configure GCLK generator 7 */
#  define CONF_CLOCK_GCLK_7_ENABLE                false
#  define CONF_CLOCK_GCLK_7_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_7_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_7_PRESCALER             1
#  define CONF_CLOCK_GCLK_7_OUTPUT_ENABLE         false

#endif /* CONF_CLOCKS_H_INCLUDED */
```

2.3.4 conf_board.h

2.3.4.1 AT32UC3A0, AT32UC3A1, AT32UC3B Devices (USBB)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

2.3.4.2 AT32UC3A3, AT32UC3A4 Devices (USBB with High Speed Support)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED
```

```
// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

2.3.4.3 AT32UC3C, ATUCXXD, ATUCXXL3U, ATUCXXL4U Devices (USBC)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

2.3.4.4 SAM3X, SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// USB pins are used
#define CONF_BOARD_USB_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

2.3.4.5 SAMD21 Device (USB)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

/* Enable USB VBUS detect */
#define CONF_BOARD_USB_VBUS_DETECT

#endif /* CONF_BOARD_H_INCLUDED */
```

3. USB Device Interface (UDI) for Human Interface Device Generic (HID Generic)

USB Device Interface (UDI) for Human Interface Device generic (HID generic) provides an interface for the configuration and management of USB HID generic device.

The outline of this documentation is as follows:

- API Overview
- Quick Start Guide for USB Device Generic Module (UDI Generic)
- Configuration File Examples

For more details for Atmel® Software Framework (ASF) USB Device Stack and USB Device HID generic, refer to following application notes:

- AVR4900: ASF - USB Device Stack[1]
- AVR4905: ASF - USB Device HID Generic Application[2]
- AVR4920: ASF - USB Device Stack - Compliance and Performance Figures[3]
- AVR4921: ASF - USB Device Stack Differences between ASF V1 and V2[4]

3.1 API Overview

3.1.1 Variable and Type Definitions

3.1.1.1 Interface with USB Device Core (UDC)

Structure required by UDC.

Variable udi_api_hid_generic

```
UDC_DESC_STORAGE udi_api_t udi_api_hid_generic
```

Global structure which contains standard UDI API for UDC.

3.1.2 Structure Definitions

3.1.2.1 Struct udi_hid_generic_desc_t

Interface descriptor structure for HID generic.

Table 3-1. Members

Type	Name	Description
usb_ep_desc_t	ep_in	Standard USB endpoint descriptor structure
usb_ep_desc_t	ep_out	Standard USB endpoint descriptor structure
usb_hid_descriptor_t	hid	HID Descriptor
usb_iface_desc_t	iface	Standard USB interface descriptor structure

[1] http://www.atmel.com/dyn/resources/prod_documents/doc8360.pdf
[2] http://www.atmel.com/dyn/resources/prod_documents/doc8499.pdf
[3] http://www.atmel.com/dyn/resources/prod_documents/doc8410.pdf
[4] http://www.atmel.com/dyn/resources/prod_documents/doc8411.pdf

3.1.2.2 Struct udi_hid_generic_report_desc_t

Report descriptor for HID generic.

Table 3-2. Members

Type	Name	Description
uint8_t	array[]	Array to put detailed report data

3.1.3 Macro Definitions

3.1.3.1 USB Interface Descriptors

The following structures provide predefined USB interface descriptors. It must be used to define the final USB descriptors.

Macro UDI_HID_GENERIC_STRING_ID

```
#define UDI_HID_GENERIC_STRING_ID 0
```

By default no string associated to this interface.

Macro UDI_HID_GENERIC_DESC

```
#define UDI_HID_GENERIC_DESC \
  {\
   .iface.bLength             = sizeof(usb_iface_desc_t),\
   .iface.bDescriptorType     = USB_DT_INTERFACE,\
   .iface.bInterfaceNumber    = UDI_HID_GENERIC_IFACE_NUMBER,\
   .iface.bAlternateSetting   = 0,\
   .iface.bNumEndpoints       = 2,\
   .iface.bInterfaceClass     = HID_CLASS,\
   .iface.bInterfaceSubClass  = HID_SUB_CLASS_NOBOOT,\
   .iface.bInterfaceProtocol  = HID_PROTOCOL_GENERIC,\
   .iface.iInterface          = UDI_HID_GENERIC_STRING_ID,\
   .hid.bLength               = sizeof(usb_hid_descriptor_t),\
   .hid.bDescriptorType       = USB_DT_HID,\
   .hid.bcdHID                = LE16(USB_HID_BDC_V1_11),\
   .hid.bCountryCode          = USB_HID_NO_COUNTRY_CODE,\
   .hid.bNumDescriptors       = USB_HID_NUM_DESC,\
   .hid.bRDescriptorType      = USB_DT_HID_REPORT,\
   .hid.wDescriptorLength     = LE16(sizeof(udi_hid_generic_report_desc_t)),\
   .ep_in.bLength             = sizeof(usb_ep_desc_t),\
   .ep_in.bDescriptorType     = USB_DT_ENDPOINT,\
   .ep_in.bEndpointAddress    = UDI_HID_GENERIC_EP_IN,\
   .ep_in.bmAttributes        = USB_EP_TYPE_INTERRUPT,\
   .ep_in.wMaxPacketSize      = LE16(UDI_HID_GENERIC_EP_SIZE),\
   .ep_in.bInterval           = 4,\
   .ep_out.bLength            = sizeof(usb_ep_desc_t),\
   .ep_out.bDescriptorType    = USB_DT_ENDPOINT,\
   .ep_out.bEndpointAddress   = UDI_HID_GENERIC_EP_OUT,\
   .ep_out.bmAttributes       = USB_EP_TYPE_INTERRUPT,\
   .ep_out.wMaxPacketSize     = LE16(UDI_HID_GENERIC_EP_SIZE),\
   .ep_out.bInterval          = 4,\
   }
```

Content of HID generic interface descriptor for all speed.

3.1.4 Function Definitions

3.1.4.1 USB Device Interface (UDI) for Human Interface Device (HID) Generic Class

Common APIs used by high level application to use this USB class.

Function udi_hid_generic_send_report_in()

Routine used to send a report to USB Host.

```
bool udi_hid_generic_send_report_in(
  uint8_t * data)
```

Table 3-3. Parameters

Data direction	Parameter name	Description
[in]	data	Pointer on the report to send (size = UDI_HID_REPORT_IN_SIZE)

Returns 1 if function was successfully done, otherwise 0.

3.2 Quick Start Guide for USB Device Generic Module (UDI Generic)

This is the quick start guide for the USB Device Generic Module (UDI Generic) with step-by-step instructions on how to configure and use the modules in a selection of use cases.

The use cases contain several code fragments. The code fragments in the steps for setup can be copied into a custom initialization function, while the steps for usage can be copied into, e.g., the main application function.

3.2.1 Basic Use Case

In this basic use case, the "USB HID generic (Single Interface Device)" module is used. The "USB HID generic (Composite Device)" module usage is described in Advanced Use Cases.

3.2.2 Setup Steps

As a USB device, it follows common USB device setup steps. Refer to USB Device Basic Setup.

3.2.3 Usage Steps

3.2.3.1 Example Code

Content of conf_usb.h:

```
#define UDI_HID_generic_ENABLE_EXT() my_callback_generic_enable()
extern bool my_callback_generic_enable(void);
#define UDI_HID_generic_DISABLE_EXT() my_callback_generic_disable()
extern void my_callback_generic_disable(void);
#include "udi_hid_generic_conf.h" // At the end of conf_usb.h file
```

Add to application C-file:

```
#define UDI_HID_GENERIC_ENABLE_EXT() my_callback_generic_enable()
extern bool my_callback_generic_enable(void);
```

```
#define UDI_HID_GENERIC_DISABLE_EXT() my_callback_generic_disable()
extern void my_callback_generic_disable(void);
#define  UDI_HID_GENERIC_REPORT_OUT(ptr) my_callback_generic_report_out(ptr)
extern void my_callback_generic_report_out(uint8_t *report);
#define  UDI_HID_GENERIC_SET_FEATURE(f) my_callback_generic_set_feature(f)
extern void my_callback_generic_set_feature(uint8_t *report_feature);

#define  UDI_HID_REPORT_IN_SIZE             64
#define  UDI_HID_REPORT_OUT_SIZE            64
#define  UDI_HID_REPORT_FEATURE_SIZE        4
#define  UDI_HID_GENERIC_EP_SIZE            64

#include "udi_hid_generic_conf.h" // At the end of conf_usb.h file
```

Add to application C-file:

```
static bool my_flag_autorize_generic_events = false;
bool my_callback_generic_enable(void)
{
   my_flag_autorize_generic_events = true;
   return true;
}
void my_callback_generic_disable(void)
{
   my_flag_autorize_generic_events = false;
}

void my_button_press_event(void)
{
   if (!my_flag_autorize_generic_events) {
      return;
   }
   uint8_t report[] = {0x00,0x01,0x02...};
   udi_hid_generic_send_report_in(report);
}

void my_callback_generic_report_out(uint8_t *report)
{
   if ((report[0] == MY_VALUE_0)
       (report[1] == MY_VALUE_1)) {
      // The report is correct
   }
}

void my_callback_generic_set_feature(uint8_t *report_feature)
{
   if ((report_feature[0] == MY_VALUE_0)
       (report_feature[1] == MY_VALUE_1)) {
      // The report feature is correct
   }
}
```

3.2.3.2 Workflow

1. Ensure that conf_usb.h is available and contains the following configuration which is the USB device generic configuration:

```
#define UDI_HID_GENERIC_ENABLE_EXT() my_callback_generic_enable()
extern bool my_callback_generic_enable(void);
```

Note After the device enumeration (detecting and identifying USB devices), the USB host starts the device configuration. When the USB generic interface from the device is accepted by the host, the USB host enables this interface and the UDI_HID_GENERIC_ENABLE_EXT() callback function is called and return true. Thus, it is recommended to enable sensors used by the generic in this function.

```
#define UDI_HID_GENERIC_DISABLE_EXT() my_callback_generic_disable()
extern void my_callback_generic_disable(void);
```

Note When the USB device is unplugged or is reset by the USB host, the USB interface is disabled and the UDI_HID_GENERIC_DISABLE_EXT() callback function is called. Thus, it is recommended to disable sensors used by the HID generic interface in this function.

```
#define  UDI_HID_GENERIC_REPORT_OUT(ptr) my_callback_generic_report_out(ptr)
extern void my_callback_generic_report_out(uint8_t *report);
```

Note Callback used to receive the OUT report.

```
#define  UDI_HID_GENERIC_SET_FEATURE(f) my_callback_generic_set_feature(f)
extern void my_callback_generic_set_feature(uint8_t *report_feature);
```

Note Callback used to receive the SET FEATURE report.

```
#define  UDI_HID_REPORT_IN_SIZE             64
#define  UDI_HID_REPORT_OUT_SIZE            64
#define  UDI_HID_REPORT_FEATURE_SIZE        4
```

Note The report size are defined by the final application.

```
#define  UDI_HID_GENERIC_EP_SIZE 64
```

Note The interrupt endpoint size is defined by the final application.

2. Send a IN report:

```
uint8_t report[] = {0x00,0x01,0x02...};
udi_hid_generic_send_report_in(report);
```

3.2.4 Advanced Use Cases

For multiple interface use of UDI HID module, see the following:

- HID Generic in a Composite Device

For more advanced use of the UDI HID generic module, see the following:

- USB Device Advanced Use Cases

3.2.5 HID Generic in a Composite Device

A USB Composite Device is a USB Device which uses more than one USB class. In this use case, the "USB HID Generic (Composite Device)" module is used to create a USB composite device. Thus, this USB module can be associated with another "Composite Device" module, like "USB MSC (Composite Device)".

Also, you can refer to application note AVR4902 ASF - USB Composite Device[5].

3.2.5.1 Setup Steps

For the setup code of this use case to work, the Basic Use Case must be followed.

3.2.5.2 Usage Steps

Example Code

Content of conf_usb.h:

```
#define USB_DEVICE_EP_CTRL_SIZE  64
#define USB_DEVICE_NB_INTERFACE (X+1)
#define USB_DEVICE_MAX_EP (X+2)

#define UDI_HID_GENERIC_EP_IN    (1 | USB_EP_DIR_IN)
#define UDI_HID_GENERIC_EP_OUT   (2 | USB_EP_DIR_OUT)
#define UDI_HID_GENERIC_IFACE_NUMBER  X

#define UDI_COMPOSITE_DESC_T \
   udi_hid_generic_desc_t udi_hid_generic; \
   ...
#define UDI_COMPOSITE_DESC_FS \
   .udi_hid_generic = UDI_HID_GENERIC_DESC, \
   ...
#define UDI_COMPOSITE_DESC_HS \
   .udi_hid_generic = UDI_HID_GENERIC_DESC, \
   ...
#define UDI_COMPOSITE_API \
   &udi_api_hid_generic, \
   ...
```

Workflow

1. Ensure that conf_usb.h is available and contains the following parameters required for a USB composite device configuration:

```
// Endpoint control size, This must be:
// - 8 for low speed
// - 8, 16, 32 or 64 for full speed device (8 is recommended to save RAM)
// - 64 for a high speed device
#define USB_DEVICE_EP_CTRL_SIZE  64
// Total Number of interfaces on this USB device.
// Add 1 for HID generic.
#define USB_DEVICE_NB_INTERFACE (X+1)
// Total number of endpoints on this USB device.
// This must include each endpoint for each interface.
// Add 1 for HID generic.
#define USB_DEVICE_MAX_EP (X+2)
```

[5] http://www.atmel.com/dyn/resources/prod_documents/doc8445.pdf

2. Ensure that conf_usb.h contains the description of composite device:

```
// The endpoint number chosen by you for the generic.
// The endpoint number starting from 1.
#define UDI_HID_GENERIC_EP_IN    (1 | USB_EP_DIR_IN)
#define UDI_HID_GENERIC_EP_OUT   (2 | USB_EP_DIR_OUT)
// The interface index of an interface starting from 0
#define UDI_HID_GENERIC_IFACE_NUMBER  X
```

3. Ensure that conf_usb.h contains the following parameters required for a USB composite device configuration:

```
// USB Interfaces descriptor structure
#define UDI_COMPOSITE_DESC_T \
   ...
   udi_hid_generic_desc_t udi_hid_generic; \
   ...
// USB Interfaces descriptor value for Full Speed
#define UDI_COMPOSITE_DESC_FS \
   ...
   .udi_hid_generic = UDI_HID_GENERIC_DESC, \
   ...
// USB Interfaces descriptor value for High Speed
#define UDI_COMPOSITE_DESC_HS \
   ...
   .udi_hid_generic = UDI_HID_GENERIC_DESC, \
   ...
// USB Interface APIs
#define UDI_COMPOSITE_API \
   ...
   &udi_api_hid_generic, \
   ...
```

Note The descriptors order given in the four lists above must be the same as the order defined by all interface indexes. The interface index orders are defined through UDI_X_IFACE_NUMBER defines.

3.3 Configuration File Examples

3.3.1 conf_usb.h

3.3.1.1 UDI HID GENERIC Single

```
#ifndef _CONF_USB_H_
#define _CONF_USB_H_

#include "compiler.h"

#warning You must refill the following definitions with a correct values

#define  USB_DEVICE_VENDOR_ID             USB_VID_ATMEL
#define  USB_DEVICE_PRODUCT_ID            USB_PID_ATMEL_ASF_HIDGENERIC
#define  USB_DEVICE_MAJOR_VERSION         1
#define  USB_DEVICE_MINOR_VERSION         0
#define  USB_DEVICE_POWER                 100 // Consumption on Vbus line (mA)
#define  USB_DEVICE_ATTR                  \
    (USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_BUS_POWERED)
```

```
//  (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_SELF_POWERED)
//  (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_BUS_POWERED)

// #define  USB_DEVICE_MANUFACTURE_NAME      "Manufacture name"
// #define  USB_DEVICE_PRODUCT_NAME          "Product name"
// #define  USB_DEVICE_SERIAL_NAME           "12...EF"

//#define  USB_DEVICE_LOW_SPEED

#if (UC3A3||UC3A4)
//#define  USB_DEVICE_HS_SUPPORT
#endif


// #define  UDC_VBUS_EVENT(b_vbus_high)      user_callback_vbus_action(b_vbus_high)
// extern void user_callback_vbus_action(bool b_vbus_high);
// #define  UDC_SOF_EVENT()                  user_callback_sof_action()
// extern void user_callback_sof_action(void);
// #define  UDC_SUSPEND_EVENT()              user_callback_suspend_action()
// extern void user_callback_suspend_action(void);
// #define  UDC_RESUME_EVENT()               user_callback_resume_action()
// extern void user_callback_resume_action(void);
// #define  UDC_REMOTEWAKEUP_ENABLE()        user_callback_remotewakeup_enable()
// extern void user_callback_remotewakeup_enable(void);
// #define  UDC_REMOTEWAKEUP_DISABLE()       user_callback_remotewakeup_disable()
// extern void user_callback_remotewakeup_disable(void);
// #define  UDC_GET_EXTRA_STRING()



#define  UDI_HID_GENERIC_ENABLE_EXT()       true
#define  UDI_HID_GENERIC_DISABLE_EXT()
#define  UDI_HID_GENERIC_REPORT_OUT(ptr)
#define  UDI_HID_GENERIC_SET_FEATURE(f)
/*
 * #define UDI_HID_GENERIC_ENABLE_EXT() my_callback_generic_enable()
 * extern bool my_callback_generic_enable(void);
 * #define UDI_HID_GENERIC_DISABLE_EXT() my_callback_generic_disable()
 * extern void my_callback_generic_disable(void);
 * #define  UDI_HID_GENERIC_REPORT_OUT(ptr) my_callback_generic_report_out(ptr)
 * extern void my_callback_generic_report_out(uint8_t *report);
 * #define  UDI_HID_GENERIC_SET_FEATURE(f) my_callback_generic_set_feature(f)
 * extern void my_callback_generic_set_feature(uint8_t *report_feature);
 */

#define  UDI_HID_REPORT_IN_SIZE             64
#define  UDI_HID_REPORT_OUT_SIZE            64
#define  UDI_HID_REPORT_FEATURE_SIZE        4

#define  UDI_HID_GENERIC_EP_SIZE            64




#include "udi_hid_generic_conf.h"

#endif // _CONF_USB_H_
```

3.3.1.2 UDI HID GENERIC Multiple (Composite)

```
#ifndef _CONF_USB_H_
#define _CONF_USB_H_

#include "compiler.h"

#warning You must refill the following definitions with a correct values

#define  USB_DEVICE_VENDOR_ID             USB_VID_ATMEL
#define  USB_DEVICE_PRODUCT_ID            0xFFFF
#define  USB_DEVICE_MAJOR_VERSION         1
#define  USB_DEVICE_MINOR_VERSION         0
#define  USB_DEVICE_POWER                 100 // Consumption on VBUS line (mA)
#define  USB_DEVICE_ATTR                  \
        (USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_BUS_POWERED)
// (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_BUS_POWERED)

// #define  USB_DEVICE_MANUFACTURE_NAME      "Manufacture name"
// #define  USB_DEVICE_PRODUCT_NAME          "Product name"
// #define  USB_DEVICE_SERIAL_NAME           "12...EF" // Disk SN for MSC

//#define  USB_DEVICE_LOW_SPEED

#if (UC3A3||UC3A4)
//#define  USB_DEVICE_HS_SUPPORT
#endif


// #define  UDC_VBUS_EVENT(b_vbus_high)      user_callback_vbus_action(b_vbus_high)
// extern void user_callback_vbus_action(bool b_vbus_high);
// #define  UDC_SOF_EVENT()                  user_callback_sof_action()
// extern void user_callback_sof_action(void);
// #define  UDC_SUSPEND_EVENT()              user_callback_suspend_action()
// extern void user_callback_suspend_action(void);
// #define  UDC_RESUME_EVENT()               user_callback_resume_action()
// extern void user_callback_resume_action(void);
// #define  UDC_REMOTEWAKEUP_ENABLE()        user_callback_remotewakeup_enable()
// extern void user_callback_remotewakeup_enable(void);
// #define  UDC_REMOTEWAKEUP_DISABLE()       user_callback_remotewakeup_disable()
// extern void user_callback_remotewakeup_disable(void);
// #define  UDC_GET_EXTRA_STRING()

#define  USB_DEVICE_EP_CTRL_SIZE       64

#define  USB_DEVICE_NB_INTERFACE       1 // 1 or more

#define  USB_DEVICE_MAX_EP             1 // 0 to max endpoint requested by interfaces




#define  UDI_CDC_PORT_NB 1

#define  UDI_CDC_ENABLE_EXT(port)             true
#define  UDI_CDC_DISABLE_EXT(port)
#define  UDI_CDC_RX_NOTIFY(port)
#define  UDI_CDC_TX_EMPTY_NOTIFY(port)
#define  UDI_CDC_SET_CODING_EXT(port,cfg)
```

```
#define  UDI_CDC_SET_DTR_EXT(port,set)
#define  UDI_CDC_SET_RTS_EXT(port,set)
/*
 * #define UDI_CDC_ENABLE_EXT(port) my_callback_cdc_enable()
 * extern bool my_callback_cdc_enable(void);
 * #define UDI_CDC_DISABLE_EXT(port) my_callback_cdc_disable()
 * extern void my_callback_cdc_disable(void);
 * #define  UDI_CDC_RX_NOTIFY(port) my_callback_rx_notify(port)
 * extern void my_callback_rx_notify(uint8_t port);
 * #define  UDI_CDC_TX_EMPTY_NOTIFY(port) my_callback_tx_empty_notify(port)
 * extern void my_callback_tx_empty_notify(uint8_t port);
 * #define  UDI_CDC_SET_CODING_EXT(port,cfg) my_callback_config(port,cfg)
 * extern void my_callback_config(uint8_t port, usb_cdc_line_coding_t * cfg);
 * #define  UDI_CDC_SET_DTR_EXT(port,set) my_callback_cdc_set_dtr(port,set)
 * extern void my_callback_cdc_set_dtr(uint8_t port, bool b_enable);
 * #define  UDI_CDC_SET_RTS_EXT(port,set) my_callback_cdc_set_rts(port,set)
 * extern void my_callback_cdc_set_rts(uint8_t port, bool b_enable);
 */

#define  UDI_CDC_LOW_RATE

#define  UDI_CDC_DEFAULT_RATE             115200
#define  UDI_CDC_DEFAULT_STOPBITS         CDC_STOP_BITS_1
#define  UDI_CDC_DEFAULT_PARITY           CDC_PAR_NONE
#define  UDI_CDC_DEFAULT_DATABITS         8

#define  UDI_CDC_DATA_EP_IN_0          (1 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_0         (2 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_0             (3 | USB_EP_DIR_IN)  // Notify endpoint
#define  UDI_CDC_DATA_EP_IN_2          (4 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_2         (5 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_2             (6 | USB_EP_DIR_IN)  // Notify endpoint
#define  UDI_CDC_DATA_EP_IN_3          (7 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_3         (8 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_3             (9 | USB_EP_DIR_IN)  // Notify endpoint

#define  UDI_CDC_COMM_IFACE_NUMBER_0   0
#define  UDI_CDC_DATA_IFACE_NUMBER_0   1
#define  UDI_CDC_COMM_IFACE_NUMBER_2   2
#define  UDI_CDC_DATA_IFACE_NUMBER_2   3
#define  UDI_CDC_COMM_IFACE_NUMBER_3   4
#define  UDI_CDC_DATA_IFACE_NUMBER_3   5




#define UDI_MSC_GLOBAL_VENDOR_ID            \
   'A', 'T', 'M', 'E', 'L', ' ', ' ', ' '
#define UDI_MSC_GLOBAL_PRODUCT_VERSION           \
   '1', '.', '0', '0'

#define  UDI_MSC_ENABLE_EXT()          true
#define  UDI_MSC_DISABLE_EXT()
#define  UDI_MSC_NOTIFY_TRANS_EXT()
/*
 * #define UDI_MSC_ENABLE_EXT() my_callback_msc_enable()
 * extern bool my_callback_msc_enable(void);
 * #define UDI_MSC_DISABLE_EXT() my_callback_msc_disable()
 * extern void my_callback_msc_disable(void);
 * #define  UDI_MSC_NOTIFY_TRANS_EXT()    msc_notify_trans()
 * extern void msc_notify_trans(void) {
```

```
 */

#define  UDI_MSC_EP_IN                 (1 | USB_EP_DIR_IN)
#define  UDI_MSC_EP_OUT                (2 | USB_EP_DIR_OUT)

#define  UDI_MSC_IFACE_NUMBER          0



#define  UDI_HID_MOUSE_ENABLE_EXT()       true
#define  UDI_HID_MOUSE_DISABLE_EXT()
// #define UDI_HID_MOUSE_ENABLE_EXT() my_callback_mouse_enable()
// extern bool my_callback_mouse_enable(void);
// #define UDI_HID_MOUSE_DISABLE_EXT() my_callback_mouse_disable()
// extern void my_callback_mouse_disable(void);

#define  UDI_HID_MOUSE_EP_IN           (1 | USB_EP_DIR_IN)

#define  UDI_HID_MOUSE_IFACE_NUMBER    0



#define  UDI_HID_KBD_ENABLE_EXT()       true
#define  UDI_HID_KBD_DISABLE_EXT()
// #define UDI_HID_KBD_ENABLE_EXT() my_callback_keyboard_enable()
// extern bool my_callback_keyboard_enable(void);
// #define UDI_HID_KBD_DISABLE_EXT() my_callback_keyboard_disable()
// extern void my_callback_keyboard_disable(void);
#define  UDI_HID_KBD_CHANGE_LED(value)
// #define  UDI_HID_KBD_CHANGE_LED(value) my_callback_keyboard_led(value)
// extern void my_callback_keyboard_led(uint8_t value)

#define  UDI_HID_KBD_EP_IN           (1 | USB_EP_DIR_IN)

#define  UDI_HID_KBD_IFACE_NUMBER    0



#define  UDI_HID_GENERIC_ENABLE_EXT()       true
#define  UDI_HID_GENERIC_DISABLE_EXT()
#define  UDI_HID_GENERIC_REPORT_OUT(ptr)
#define  UDI_HID_GENERIC_SET_FEATURE(f)
/*
 * #define UDI_HID_GENERIC_ENABLE_EXT() my_callback_generic_enable()
 * extern bool my_callback_generic_enable(void);
 * #define UDI_HID_GENERIC_DISABLE_EXT() my_callback_generic_disable()
 * extern void my_callback_generic_disable(void);
 * #define  UDI_HID_GENERIC_REPORT_OUT(ptr) my_callback_generic_report_out(ptr)
 * extern void my_callback_generic_report_out(uint8_t *report);
 * #define  UDI_HID_GENERIC_SET_FEATURE(f) my_callback_generic_set_feature(f)
 * extern void my_callback_generic_set_feature(uint8_t *report_feature);
 */
#define  UDI_HID_REPORT_IN_SIZE             64
#define  UDI_HID_REPORT_OUT_SIZE            64
#define  UDI_HID_REPORT_FEATURE_SIZE        4
#define  UDI_HID_GENERIC_EP_SIZE            64

#define  UDI_HID_GENERIC_EP_OUT   (2 | USB_EP_DIR_OUT)
```

```
#define  UDI_HID_GENERIC_EP_IN    (1 | USB_EP_DIR_IN)

#define  UDI_HID_GENERIC_IFACE_NUMBER    0



#define  UDI_PHDC_ENABLE_EXT()          true
#define  UDI_PHDC_DISABLE_EXT()

#define  UDI_PHDC_DATAMSG_FORMAT       USB_PHDC_DATAMSG_FORMAT_11073_20601
#define  UDI_PHDC_SPECIALIZATION       {0x2345} // Define in 11073_20601

#define  UDI_PHDC_QOS_OUT              \
        (USB_PHDC_QOS_MEDIUM_BETTER|USB_PHDC_QOS_HIGH_BEST)
#define  UDI_PHDC_QOS_IN               \
        (USB_PHDC_QOS_LOW_GOOD|USB_PHDC_QOS_MEDIUM_BETTER|USB_PHDC_QOS_MEDIUM_BEST)

#define  UDI_PHDC_METADATA_DESC_BULK_IN   {0x01,0x02,0x03}
#define  UDI_PHDC_METADATA_DESC_BULK_OUT  {0x01,0x02,0x03}
#define  UDI_PHDC_METADATA_DESC_INT_IN    {0x01,0x02,0x03}

#define  UDI_PHDC_EP_BULK_OUT          (1 | USB_EP_DIR_OUT)
#define  UDI_PHDC_EP_BULK_IN           (2 | USB_EP_DIR_IN)
#if ((UDI_PHDC_QOS_IN&USB_PHDC_QOS_LOW_GOOD)==USB_PHDC_QOS_LOW_GOOD)
// Only if UDI_PHDC_QOS_IN include USB_PHDC_QOS_LOW_GOOD
#  define  UDI_PHDC_EP_INTERRUPT_IN      (3 | USB_EP_DIR_IN)
#endif

#define  UDI_PHDC_EP_SIZE_BULK_OUT     32
#define  UDI_PHDC_EP_SIZE_BULK_IN      32
#define  UDI_PHDC_EP_SIZE_INT_IN       8


#define  UDI_PHDC_IFACE_NUMBER          0



#define  UDI_VENDOR_ENABLE_EXT()          true
#define  UDI_VENDOR_DISABLE_EXT()
#define  UDI_VENDOR_SETUP_OUT_RECEIVED()  false
#define  UDI_VENDOR_SETUP_IN_RECEIVED()   false
/*
 * #define UDI_VENDOR_ENABLE_EXT() my_callback_vendor_enable()
 * extern bool my_callback_vendor_enable(void);
 * #define UDI_VENDOR_DISABLE_EXT() my_callback_vendor_disable()
 * extern void my_callback_vendor_disable(void);
 *
 * #define  UDI_VENDOR_SETUP_OUT_RECEIVED()  my_vendor_setup_out_received()
 * extern bool my_vendor_setup_out_received(void);
 * #define  UDI_VENDOR_SETUP_IN_RECEIVED()   my_vendor_setup_in_received()
 * extern bool my_vendor_setup_in_received(void);
 */

#define  UDI_VENDOR_EPS_SIZE_INT_FS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_FS   64
#define  UDI_VENDOR_EPS_SIZE_ISO_FS   256

#define  UDI_VENDOR_EPS_SIZE_INT_HS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_HS  512
#define  UDI_VENDOR_EPS_SIZE_ISO_HS    64
```

```
#define  UDI_VENDOR_EP_INTERRUPT_IN  (1 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_INTERRUPT_OUT (2 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_BULK_IN       (3 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_BULK_OUT      (4 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_ISO_IN        (5 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_ISO_OUT       (6 | USB_EP_DIR_OUT)


#define  UDI_VENDOR_IFACE_NUMBER     0



//... Eventually add other Interface Configuration



#define UDI_COMPOSITE_DESC_T

#define UDI_COMPOSITE_DESC_FS

#define UDI_COMPOSITE_DESC_HS

#define UDI_COMPOSITE_API

/* Example for device with cdc, msc and hid mouse interface
#define UDI_COMPOSITE_DESC_T \
    usb_iad_desc_t udi_cdc_iad; \
    udi_cdc_comm_desc_t udi_cdc_comm; \
    udi_cdc_data_desc_t udi_cdc_data; \
    udi_msc_desc_t udi_msc; \
    udi_hid_mouse_desc_t udi_hid_mouse

#define UDI_COMPOSITE_DESC_FS \
    .udi_cdc_iad               = UDI_CDC_IAD_DESC_0, \
    .udi_cdc_comm              = UDI_CDC_COMM_DESC_0, \
    .udi_cdc_data              = UDI_CDC_DATA_DESC_0_FS, \
    .udi_msc                   = UDI_MSC_DESC_FS, \
    .udi_hid_mouse             = UDI_HID_MOUSE_DESC

#define UDI_COMPOSITE_DESC_HS \
    .udi_cdc_iad               = UDI_CDC_IAD_DESC_0, \
    .udi_cdc_comm              = UDI_CDC_COMM_DESC_0, \
    .udi_cdc_data              = UDI_CDC_DATA_DESC_0_HS, \
    .udi_msc                   = UDI_MSC_DESC_HS, \
    .udi_hid_mouse             = UDI_HID_MOUSE_DESC

#define UDI_COMPOSITE_API  \
    &udi_api_cdc_comm,       \
    &udi_api_cdc_data,       \
    &udi_api_msc,            \
    &udi_api_hid_mouse
*/




/* Example of include for interface
#include "udi_msc.h"
#include "udi_hid_kbd.h"
```

```
#include "udi_hid_mouse.h"
#include "udi_cdc.h"
#include "udi_phdc.h"
#include "udi_vendor.h"
*/
/* Declaration of callbacks used by USB
#include "callback_def.h"
*/

#endif // _CONF_USB_H_
```

3.3.2 conf_clock.h

3.3.2.1 AT32UC3C, ATUCXXD, ATUCXXL3U, ATUCXXL4U Devices (USBC)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC1
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_PLL0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL1
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RC8M

// ===== PLL0 Options
#define CONFIG_PLL0_SOURCE            PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_SOURCE          PLL_SRC_RC8M
#define CONFIG_PLL0_MUL               3 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL0_DIV               1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL1_SOURCE          PLL_SRC_RC8M
//#define CONFIG_PLL1_MUL             3 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL1_DIV             1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
//#define CONFIG_SYSCLK_CPU_DIV       0 /* Fcpu = Fsys/(2 ^ CPU_div) */
//#define CONFIG_SYSCLK_PBA_DIV       0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV       0 /* Fpbb = Fsys/(2 ^ PBB_div) */
//#define CONFIG_SYSCLK_PBC_DIV       0 /* Fpbc = Fsys/(2 ^ PBC_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC1
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

3.3.2.2 SAM3X, SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock (MCK) Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_BYPASS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_4M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_8M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_12M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_BYPASS
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLLACK
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_UPLLCK

// ===== System Clock (MCK) Prescaler Options   (Fmck = Fsys / (SYSCLK_PRES))
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_1
#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_2
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_4
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_8
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_16
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_32
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_64
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_3

// ===== PLL0 (A) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
// Use mul and div effective values here.
#define CONFIG_PLL0_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL0_MUL             14
#define CONFIG_PLL0_DIV             1

// ===== UPLL (UTMI) Hardware fixed at 480MHz.

// ===== USB Clock Source Options   (Fusb = FpllX / USB_div)
// Use div effective value here.
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_UPLL
#define CONFIG_USBCLK_DIV           1

// ===== Target frequency (System clock)
// - XTAL frequency: 12MHz
// - System clock source: PLLA
// - System clock prescaler: 2 (divided by 2)
// - PLLA source: XTAL
// - PLLA output: XTAL * 14 / 1
// - System clock is: 12 * 14 / 1 /2 = 84MHz
// ===== Target frequency (USB Clock)
// - USB clock source: UPLL
// - USB clock divider: 1 (not divided)
// - UPLL frequency: 480MHz
// - USB clock: 480 / 1 = 480MHz


#endif /* CONF_CLOCK_H_INCLUDED */
```

3.3.3 conf_clocks.h

3.3.3.1 SAMD21 Device (USB)

```
#include <clock.h>

#ifndef CONF_CLOCKS_H_INCLUDED
#  define CONF_CLOCKS_H_INCLUDED

/* System clock bus configuration */
#  define CONF_CLOCK_CPU_CLOCK_FAILURE_DETECT     false
#  define CONF_CLOCK_FLASH_WAIT_STATES            2
#  define CONF_CLOCK_CPU_DIVIDER                  SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBA_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBB_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1

/* SYSTEM_CLOCK_SOURCE_OSC8M configuration - Internal 8MHz oscillator */
#  define CONF_CLOCK_OSC8M_PRESCALER              SYSTEM_OSC8M_DIV_1
#  define CONF_CLOCK_OSC8M_ON_DEMAND              true
#  define CONF_CLOCK_OSC8M_RUN_IN_STANDBY         false

/* SYSTEM_CLOCK_SOURCE_XOSC configuration - External clock/oscillator */
#  define CONF_CLOCK_XOSC_ENABLE                  false
#  define CONF_CLOCK_XOSC_EXTERNAL_CRYSTAL        SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC_EXTERNAL_FREQUENCY      12000000UL
#  define CONF_CLOCK_XOSC_STARTUP_TIME            SYSTEM_XOSC_STARTUP_32768
#  define CONF_CLOCK_XOSC_AUTO_GAIN_CONTROL       true
#  define CONF_CLOCK_XOSC_ON_DEMAND               true
#  define CONF_CLOCK_XOSC_RUN_IN_STANDBY          false

/* SYSTEM_CLOCK_SOURCE_XOSC32K configuration - External 32KHz crystal/clock oscillator */
#  define CONF_CLOCK_XOSC32K_ENABLE               false
#  define CONF_CLOCK_XOSC32K_EXTERNAL_CRYSTAL     SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC32K_STARTUP_TIME         SYSTEM_XOSC32K_STARTUP_65536
#  define CONF_CLOCK_XOSC32K_AUTO_AMPLITUDE_CONTROL  false
#  define CONF_CLOCK_XOSC32K_ENABLE_1KHZ_OUPUT    false
#  define CONF_CLOCK_XOSC32K_ENABLE_32KHZ_OUTPUT  true
#  define CONF_CLOCK_XOSC32K_ON_DEMAND            true
#  define CONF_CLOCK_XOSC32K_RUN_IN_STANDBY       false

/* SYSTEM_CLOCK_SOURCE_OSC32K configuration - Internal 32KHz oscillator */
#  define CONF_CLOCK_OSC32K_ENABLE                false
#  define CONF_CLOCK_OSC32K_STARTUP_TIME          SYSTEM_OSC32K_STARTUP_130
#  define CONF_CLOCK_OSC32K_ENABLE_1KHZ_OUTPUT    true
#  define CONF_CLOCK_OSC32K_ENABLE_32KHZ_OUTPUT   true
#  define CONF_CLOCK_OSC32K_ON_DEMAND             true
#  define CONF_CLOCK_OSC32K_RUN_IN_STANDBY        false

/* SYSTEM_CLOCK_SOURCE_DFLL configuration - Digital Frequency Locked Loop */
#  define CONF_CLOCK_DFLL_ENABLE                  true
#  define CONF_CLOCK_DFLL_LOOP_MODE               SYSTEM_CLOCK_DFLL_LOOP_MODE_USB_RECOVERY
#  define CONF_CLOCK_DFLL_ON_DEMAND               true

/* DFLL open loop mode configuration */
#  define CONF_CLOCK_DFLL_COARSE_VALUE            (0x1f / 4)
#  define CONF_CLOCK_DFLL_FINE_VALUE              (0xff / 4)

/* DFLL closed loop mode configuration */
#  define CONF_CLOCK_DFLL_SOURCE_GCLK_GENERATOR   GCLK_GENERATOR_1
#  define CONF_CLOCK_DFLL_MULTIPLY_FACTOR         (48000000 / 32768)
#  define CONF_CLOCK_DFLL_QUICK_LOCK              true
```

```
#  define CONF_CLOCK_DFLL_TRACK_AFTER_FINE_LOCK   true
#  define CONF_CLOCK_DFLL_KEEP_LOCK_ON_WAKEUP     true
#  define CONF_CLOCK_DFLL_ENABLE_CHILL_CYCLE      true
#  define CONF_CLOCK_DFLL_MAX_COARSE_STEP_SIZE    (0x1f / 4)
#  define CONF_CLOCK_DFLL_MAX_FINE_STEP_SIZE      (0xff / 4)

/* SYSTEM_CLOCK_SOURCE_DPLL configuration - Digital Phase-Locked Loop */
#  define CONF_CLOCK_DPLL_ENABLE                  false
#  define CONF_CLOCK_DPLL_ON_DEMAND               true
#  define CONF_CLOCK_DPLL_RUN_IN_STANDBY          false
#  define CONF_CLOCK_DPLL_LOCK_BYPASS             false
#  define CONF_CLOCK_DPLL_WAKE_UP_FAST            false
#  define CONF_CLOCK_DPLL_LOW_POWER_ENABLE        false

#  define CONF_CLOCK_DPLL_LOCK_TIME               SYSTEM_CLOCK_SOURCE_DPLL_LOCK_TIME_NO_TIMEOUT
#  define CONF_CLOCK_DPLL_REFERENCE_CLOCK         SYSTEM_CLOCK_SOURCE_DPLL_REFERENCE_CLOCK_REF0
#  define CONF_CLOCK_DPLL_FILTER                  SYSTEM_CLOCK_SOURCE_DPLL_FILTER_DEFAULT

#  define CONF_CLOCK_DPLL_REFERENCE_FREQUENCY     32768
#  define CONF_CLOCK_DPLL_REFEREMCE_DIVIDER       1
#  define CONF_CLOCK_DPLL_OUTPUT_FREQUENCY        48000000

/* Set this to true to configure the GCLK when running clocks_init. If set to
 * false, none of the GCLK generators will be configured in clocks_init(). */
#  define CONF_CLOCK_CONFIGURE_GCLK               true

/* Configure GCLK generator 0 (Main Clock) */
#  define CONF_CLOCK_GCLK_0_ENABLE                true
#  define CONF_CLOCK_GCLK_0_RUN_IN_STANDBY        true
#  define CONF_CLOCK_GCLK_0_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_DFLL
#  define CONF_CLOCK_GCLK_0_PRESCALER             1
#  define CONF_CLOCK_GCLK_0_OUTPUT_ENABLE         false

/* Configure GCLK generator 1 */
#  define CONF_CLOCK_GCLK_1_ENABLE                false
#  define CONF_CLOCK_GCLK_1_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_1_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_XOSC32K
#  define CONF_CLOCK_GCLK_1_PRESCALER             1
#  define CONF_CLOCK_GCLK_1_OUTPUT_ENABLE         false

/* Configure GCLK generator 2 (RTC) */
#  define CONF_CLOCK_GCLK_2_ENABLE                false
#  define CONF_CLOCK_GCLK_2_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_2_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC32K
#  define CONF_CLOCK_GCLK_2_PRESCALER             32
#  define CONF_CLOCK_GCLK_2_OUTPUT_ENABLE         false

/* Configure GCLK generator 3 */
#  define CONF_CLOCK_GCLK_3_ENABLE                false
#  define CONF_CLOCK_GCLK_3_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_3_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_3_PRESCALER             1
#  define CONF_CLOCK_GCLK_3_OUTPUT_ENABLE         false

/* Configure GCLK generator 4 */
#  define CONF_CLOCK_GCLK_4_ENABLE                false
#  define CONF_CLOCK_GCLK_4_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_4_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_4_PRESCALER             1
#  define CONF_CLOCK_GCLK_4_OUTPUT_ENABLE         false
```

```
/* Configure GCLK generator 5 */
#  define CONF_CLOCK_GCLK_5_ENABLE                 false
#  define CONF_CLOCK_GCLK_5_RUN_IN_STANDBY         false
#  define CONF_CLOCK_GCLK_5_CLOCK_SOURCE           SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_5_PRESCALER              1
#  define CONF_CLOCK_GCLK_5_OUTPUT_ENABLE          false

/* Configure GCLK generator 6 */
#  define CONF_CLOCK_GCLK_6_ENABLE                 false
#  define CONF_CLOCK_GCLK_6_RUN_IN_STANDBY         false
#  define CONF_CLOCK_GCLK_6_CLOCK_SOURCE           SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_6_PRESCALER              1
#  define CONF_CLOCK_GCLK_6_OUTPUT_ENABLE          false

/* Configure GCLK generator 7 */
#  define CONF_CLOCK_GCLK_7_ENABLE                 false
#  define CONF_CLOCK_GCLK_7_RUN_IN_STANDBY         false
#  define CONF_CLOCK_GCLK_7_CLOCK_SOURCE           SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_7_PRESCALER              1
#  define CONF_CLOCK_GCLK_7_OUTPUT_ENABLE          false

#endif /* CONF_CLOCKS_H_INCLUDED */
```

3.3.4 conf_board.h

3.3.4.1 AT32UC3C, ATUCXXD, ATUCXXL3U, ATUCXXL4U Devices (USBC)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

3.3.4.2 SAM3X, SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// USB pins are used
#define CONF_BOARD_USB_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

3.3.4.3 SAMD21 Device (USB)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

/* Enable USB VBUS detect */
#define CONF_BOARD_USB_VBUS_DETECT

#endif /* CONF_BOARD_H_INCLUDED */
```

4. USB Device Interface (UDI) for Human Interface Device Keyboard (HID Keyboard)

USB Device Interface (UDI) for Human Interface Device Keyboard (HID keyboard) provides an interface for the configuration and management of USB HID keyboard device.

The outline of this documentation is as follows:

- API Overview
- Quick Start Guide for USB Device Keyboard Module (UDI Keyboard)
- Configuration File Examples

For more details for Atmel® Software Framework (ASF) USB Device Stack and USB Device HID keyboard, refer to following application notes:

- AVR4900: ASF - USB Device Stack[1]
- AVR4904: ASF - USB Device HID Keyboard Application[2]
- AVR4920: ASF - USB Device Stack - Compliance and Performance Figures[3]
- AVR4921: ASF - USB Device Stack Differences between ASF V1 and V2[4]

4.1 API Overview

4.1.1 Variable and Type Definitions

4.1.1.1 Interface with USB Device Core (UDC)

Variable required by UDC.

Variable udi_api_hid_kbd

```
UDC_DESC_STORAGE udi_api_t udi_api_hid_kbd
```

Global structure which contains standard UDI API for UDC.

4.1.2 Structure Definitions

4.1.2.1 Struct udi_hid_kbd_desc_t

Interface descriptor structure for HID keyboard.

Table 4-1. Members

Type	Name	Description
usb_ep_desc_t	ep	Standard USB endpoint descriptor structure
usb_hid_descriptor_t	hid	HID Descriptor
usb_iface_desc_t	iface	Standard USB interface descriptor structure

[1] http://www.atmel.com/dyn/resources/prod_documents/doc8360.pdf
[2] http://www.atmel.com/dyn/resources/prod_documents/doc8446.pdf
[3] http://www.atmel.com/dyn/resources/prod_documents/doc8410.pdf
[4] http://www.atmel.com/dyn/resources/prod_documents/doc8411.pdf

4.1.2.2 Struct udi_hid_kbd_report_desc_t

Report descriptor for HID keyboard.

Table 4-2. Members

Type	Name	Description
uint8_t	array[]	Array to put detailed report data

4.1.3 Macro Definitions

4.1.3.1 USB Interface Descriptors

The following structures provide predefined USB interface descriptors. It must be used to define the final USB descriptors.

Macro UDI_HID_KBD_STRING_ID

```
#define UDI_HID_KBD_STRING_ID 0
```

By default no string associated to this interface.

Macro UDI_HID_KBD_EP_SIZE

```
#define UDI_HID_KBD_EP_SIZE 8
```

HID keyboard endpoints size.

Macro UDI_HID_KBD_DESC

```
#define UDI_HID_KBD_DESC \
  {\
 .iface.bLength             = sizeof(usb_iface_desc_t),\
 .iface.bDescriptorType     = USB_DT_INTERFACE,\
 .iface.bInterfaceNumber    = UDI_HID_KBD_IFACE_NUMBER,\
 .iface.bAlternateSetting   = 0,\
 .iface.bNumEndpoints       = 1,\
 .iface.bInterfaceClass     = HID_CLASS,\
 .iface.bInterfaceSubClass  = HID_SUB_CLASS_NOBOOT,\
 .iface.bInterfaceProtocol  = HID_PROTOCOL_KEYBOARD,\
 .iface.iInterface          = UDI_HID_KBD_STRING_ID,\
 .hid.bLength               = sizeof(usb_hid_descriptor_t),\
 .hid.bDescriptorType       = USB_DT_HID,\
 .hid.bcdHID                = LE16(USB_HID_BDC_V1_11),\
 .hid.bCountryCode          = USB_HID_NO_COUNTRY_CODE,\
 .hid.bNumDescriptors       = USB_HID_NUM_DESC,\
 .hid.bRDescriptorType      = USB_DT_HID_REPORT,\
 .hid.wDescriptorLength     = LE16(sizeof(udi_hid_kbd_report_desc_t)),\
 .ep.bLength                = sizeof(usb_ep_desc_t),\
 .ep.bDescriptorType        = USB_DT_ENDPOINT,\
 .ep.bEndpointAddress       = UDI_HID_KBD_EP_IN,\
 .ep.bmAttributes           = USB_EP_TYPE_INTERRUPT,\
 .ep.wMaxPacketSize         = LE16(UDI_HID_KBD_EP_SIZE),\
 .ep.bInterval              = 2,\
```

```
}
```

Content of HID keyboard interface descriptor for all speed.

4.1.4 Function Definitions

4.1.4.1 USB Device Interface (UDI) for Human Interface Device (HID) Keyboard Class

Common APIs used by high level application to use this USB class.

Function udi_hid_kbd_modifier_up()

Send events key modifier released.

```
bool udi_hid_kbd_modifier_up(
  uint8_t modifier_id)
```

Table 4-3. Parameters

Data direction	Parameter name	Description
[in]	modifier_id	ID of key modifier

Returns 1 if function was successfully done, otherwise 0.

Function udi_hid_kbd_modifier_down()

Send events key modifier pressed.

```
bool udi_hid_kbd_modifier_down(
  uint8_t modifier_id)
```

Table 4-4. Parameters

Data direction	Parameter name	Description
[in]	modifier_id	ID of key modifier

Returns 1 if function was successfully done, otherwise 0.

Function udi_hid_kbd_up()

Send events key released.

```
bool udi_hid_kbd_up(
  uint8_t key_id)
```

Table 4-5. Parameters

Data direction	Parameter name	Description
[in]	key_id	ID of key

Returns 1 if function was successfully done, otherwise 0.

Function udi_hid_kbd_down()

Send events key pressed.

```
bool udi_hid_kbd_down(
  uint8_t key_id)
```

Table 4-6. Parameters

Data direction	Parameter name	Description
[in]	key_id	ID of key

Returns 1 if function was successfully done, otherwise 0.

4.2 Quick Start Guide for USB Device Keyboard Module (UDI Keyboard)

This is the quick start guide for the USB Device Keyboard Module (UDI Keyboard) with step-by-step instructions on how to configure and use the modules in a selection of use cases.

The use cases contain several code fragments. The code fragments in the steps for setup can be copied into a custom initialization function, while the steps for usage can be copied into, e.g., the main application function.

4.2.1 Basic Use Case

In this basic use case, the "USB HID keyboard (Single Interface Device)" module is used. The "USB HID keyboard (Composite Device)" module usage is described in Advanced Use Cases.

4.2.2 Setup Steps

As a USB device, it follows common USB device setup steps. Refer to USB Device Basic Setup.

4.2.3 Usage Steps

4.2.3.1 Example Code

Content of conf_usb.h:

```
#define UDI_HID_KBD_ENABLE_EXT() my_callback_keyboard_enable()
extern bool my_callback_keyboard_enable(void);
#define UDI_HID_KBD_DISABLE_EXT() my_callback_keyboard_disable()
extern void my_callback_keyboard_disable(void);
#include "udi_hid_keyboard_conf.h" // At the end of conf_usb.h file
```

Add to application C-file:

```
static bool my_flag_autorize_keyboard_events = false;
bool my_callback_keyboard_enable(void)
{
   my_flag_autorize_keyboard_events = true;
   return true;
}
```

```
void my_callback_keyboard_disable(void)
{
   my_flag_autorize_keyboard_events = false;
}

void my_key_A_press_event(void)
{
   if (!my_flag_autorize_keyboard_events) {
      return;
   }
   udi_hid_kbd_up(HID_A);
}
```

4.2.3.2 Workflow

1. Ensure that conf_usb.h is available and contains the following configuration which is the USB device keyboard configuration:

```
#define UDI_HID_KBD_ENABLE_EXT() my_callback_keyboard_enable()
extern bool my_callback_keyboard_enable(void);
```

Note After the device enumeration (detecting and identifying USB devices), the USB host starts the device configuration. When the USB keyboard interface from the device is accepted by the host, the USB host enables this interface and the UDI_HID_KBD_ENABLE_EXT() callback function is called and return true. Thus, it is recommended to enable sensors used by the keyboard in this function.

```
#define UDI_HID_KBD_DISABLE_EXT() my_callback_keyboard_disable()
extern void my_callback_keyboard_disable(void);
```

Note When the USB device is unplugged or is reset by the USB host, the USB interface is disabled and the UDI_HID_KBD_DISABLE_EXT() callback function is called. Thus, it is recommended to disable sensors used by the keyboard in this function.

2. Send keyboard events:

```
// Send events key modifier released
udi_hid_kbd_modifier_up(uint8_t modifier_id);
// Send events key modifier pressed
udi_hid_kbd_modifier_down(uint8_t modifier_id);
// Send events key released
udi_hid_kbd_up(uint8_t key_id);
// Send events key pressed
udi_hid_kbd_down(uint8_t key_id);
```

4.2.4 Advanced Use Cases

For multiple interface use of UDI HID module, see the following:

- HID Keyboard in a Composite Device

For more advanced use of the UDI HID keyboard module, see the following:

- USB Device Advanced Use Cases

4.2.5 HID Keyboard in a Composite Device

A USB Composite Device is a USB Device which uses more than one USB class. In this use case, the "USB HID Keyboard (Composite Device)" module is used to create a USB composite device. Thus, this USB module can be associated with another "Composite Device" module, like "USB MSC (Composite Device)".

Also, you can refer to application note AVR4902 ASF - USB Composite Device[5].

4.2.5.1 Setup Steps

For the setup code of this use case to work, the Basic Use Case must be followed.

4.2.5.2 Usage Steps

Example Code

Content of conf_usb.h:

```
#define USB_DEVICE_EP_CTRL_SIZE  64
#define USB_DEVICE_NB_INTERFACE (X+1)
#define USB_DEVICE_MAX_EP (X+1)

#define UDI_HID_KBD_EP_IN  (X | USB_EP_DIR_IN)
#define UDI_HID_KBD_IFACE_NUMBER  X

#define UDI_COMPOSITE_DESC_T \
   udi_hid_kbd_desc_t udi_hid_kbd; \
   ...
#define UDI_COMPOSITE_DESC_FS \
   .udi_hid_kbd = UDI_HID_KBD_DESC, \
   ...
#define UDI_COMPOSITE_DESC_HS \
   .udi_hid_kbd = UDI_HID_KBD_DESC, \
   ...
#define UDI_COMPOSITE_API \
   &udi_api_hid_kbd, \
   ...
```

Workflow

1. Ensure that conf_usb.h is available and contains the following parameters required for a USB composite device configuration:

```
// Endpoint control size, This must be:
// - 8 for low speed
// - 8, 16, 32 or 64 for full speed device (8 is recommended to save RAM)
// - 64 for a high speed device
#define USB_DEVICE_EP_CTRL_SIZE  64
// Total Number of interfaces on this USB device.
// Add 1 for HID keyboard.
#define USB_DEVICE_NB_INTERFACE (X+1)
// Total number of endpoints on this USB device.
// This must include each endpoint for each interface.
// Add 1 for HID keyboard.
#define USB_DEVICE_MAX_EP (X+1)
```

2. Ensure that conf_usb.h contains the description of composite device:

```
// The endpoint number chosen by you for the keyboard.
```

[5] http://www.atmel.com/dyn/resources/prod_documents/doc8445.pdf

```
// The endpoint number starting from 1.
#define UDI_HID_KBD_EP_IN  (X | USB_EP_DIR_IN)
// The interface index of an interface starting from 0
#define UDI_HID_KBD_IFACE_NUMBER  X
```

3. Ensure that conf_usb.h contains the following parameters required for a USB composite device configuration:

```
// USB Interfaces descriptor structure
#define UDI_COMPOSITE_DESC_T \
   ...
   udi_hid_kbd_desc_t udi_hid_kbd; \
   ...
// USB Interfaces descriptor value for Full Speed
#define UDI_COMPOSITE_DESC_FS \
   ...
   .udi_hid_kbd = UDI_HID_KBD_DESC, \
   ...
// USB Interfaces descriptor value for High Speed
#define UDI_COMPOSITE_DESC_HS \
   ...
   .udi_hid_kbd = UDI_HID_KBD_DESC, \
   ...
// USB Interface APIs
#define UDI_COMPOSITE_API \
   ...
   &udi_api_hid_kbd, \
      ...
```

Note The descriptors order given in the four lists above must be the same as the order defined by all interface indexes. The interface index orders are defined through UDI_X_IFACE_NUMBER defines.

4.3 Configuration File Examples

4.3.1 conf_usb.h

4.3.1.1 UDI HID KBD Single

```
#ifndef _CONF_USB_H_
#define _CONF_USB_H_

#include "compiler.h"

#warning You must refill the following definitions with a correct values

#define  USB_DEVICE_VENDOR_ID             USB_VID_ATMEL
#define  USB_DEVICE_PRODUCT_ID            USB_PID_ATMEL_ASF_HIDGENERIC
#define  USB_DEVICE_MAJOR_VERSION         1
#define  USB_DEVICE_MINOR_VERSION         0
#define  USB_DEVICE_POWER                 100 // Consumption on Vbus line (mA)
#define  USB_DEVICE_ATTR                  \
    (USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_BUS_POWERED)
//  (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_SELF_POWERED)
//  (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_BUS_POWERED)

// #define  USB_DEVICE_MANUFACTURE_NAME      "Manufacture name"
```

```
// #define  USB_DEVICE_PRODUCT_NAME          "Product name"
// #define  USB_DEVICE_SERIAL_NAME           "12...EF"

//#define  USB_DEVICE_LOW_SPEED

#if (UC3A3||UC3A4)
//#define  USB_DEVICE_HS_SUPPORT
#endif


// #define  UDC_VBUS_EVENT(b_vbus_high)      user_callback_vbus_action(b_vbus_high)
// extern void user_callback_vbus_action(bool b_vbus_high);
// #define  UDC_SOF_EVENT()                  user_callback_sof_action()
// extern void user_callback_sof_action(void);
// #define  UDC_SUSPEND_EVENT()              user_callback_suspend_action()
// extern void user_callback_suspend_action(void);
// #define  UDC_RESUME_EVENT()               user_callback_resume_action()
// extern void user_callback_resume_action(void);
// #define  UDC_REMOTEWAKEUP_ENABLE()        user_callback_remotewakeup_enable()
// extern void user_callback_remotewakeup_enable(void);
// #define  UDC_REMOTEWAKEUP_DISABLE()       user_callback_remotewakeup_disable()
// extern void user_callback_remotewakeup_disable(void);
// #define  UDC_GET_EXTRA_STRING()



#define  UDI_HID_GENERIC_ENABLE_EXT()       true
#define  UDI_HID_GENERIC_DISABLE_EXT()
#define  UDI_HID_GENERIC_REPORT_OUT(ptr)
#define  UDI_HID_GENERIC_SET_FEATURE(f)
/*
 * #define UDI_HID_GENERIC_ENABLE_EXT() my_callback_generic_enable()
 * extern bool my_callback_generic_enable(void);
 * #define UDI_HID_GENERIC_DISABLE_EXT() my_callback_generic_disable()
 * extern void my_callback_generic_disable(void);
 * #define  UDI_HID_GENERIC_REPORT_OUT(ptr) my_callback_generic_report_out(ptr)
 * extern void my_callback_generic_report_out(uint8_t *report);
 * #define  UDI_HID_GENERIC_SET_FEATURE(f) my_callback_generic_set_feature(f)
 * extern void my_callback_generic_set_feature(uint8_t *report_feature);
 */

#define  UDI_HID_REPORT_IN_SIZE             64
#define  UDI_HID_REPORT_OUT_SIZE            64
#define  UDI_HID_REPORT_FEATURE_SIZE        4

#define  UDI_HID_GENERIC_EP_SIZE            64




#include "udi_hid_generic_conf.h"

#endif // _CONF_USB_H_
```

4.3.1.2 UDI HID KBD Multiple (Composite)

```
#ifndef _CONF_USB_H_
#define _CONF_USB_H_
```

```
#include "compiler.h"

#warning You must refill the following definitions with a correct values

#define  USB_DEVICE_VENDOR_ID             USB_VID_ATMEL
#define  USB_DEVICE_PRODUCT_ID            0xFFFF
#define  USB_DEVICE_MAJOR_VERSION         1
#define  USB_DEVICE_MINOR_VERSION         0
#define  USB_DEVICE_POWER                 100 // Consumption on VBUS line (mA)
#define  USB_DEVICE_ATTR                  \
        (USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_BUS_POWERED)
// (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_BUS_POWERED)

// #define  USB_DEVICE_MANUFACTURE_NAME      "Manufacture name"
// #define  USB_DEVICE_PRODUCT_NAME          "Product name"
// #define  USB_DEVICE_SERIAL_NAME           "12...EF" // Disk SN for MSC

//#define  USB_DEVICE_LOW_SPEED

#if (UC3A3||UC3A4)
//#define  USB_DEVICE_HS_SUPPORT
#endif


// #define  UDC_VBUS_EVENT(b_vbus_high)      user_callback_vbus_action(b_vbus_high)
// extern void user_callback_vbus_action(bool b_vbus_high);
// #define  UDC_SOF_EVENT()                  user_callback_sof_action()
// extern void user_callback_sof_action(void);
// #define  UDC_SUSPEND_EVENT()              user_callback_suspend_action()
// extern void user_callback_suspend_action(void);
// #define  UDC_RESUME_EVENT()               user_callback_resume_action()
// extern void user_callback_resume_action(void);
// #define  UDC_REMOTEWAKEUP_ENABLE()        user_callback_remotewakeup_enable()
// extern void user_callback_remotewakeup_enable(void);
// #define  UDC_REMOTEWAKEUP_DISABLE()       user_callback_remotewakeup_disable()
// extern void user_callback_remotewakeup_disable(void);
// #define  UDC_GET_EXTRA_STRING()

#define  USB_DEVICE_EP_CTRL_SIZE       64

#define  USB_DEVICE_NB_INTERFACE       1 // 1 or more

#define  USB_DEVICE_MAX_EP             1 // 0 to max endpoint requested by interfaces



#define  UDI_CDC_PORT_NB 1

#define  UDI_CDC_ENABLE_EXT(port)             true
#define  UDI_CDC_DISABLE_EXT(port)
#define  UDI_CDC_RX_NOTIFY(port)
#define  UDI_CDC_TX_EMPTY_NOTIFY(port)
#define  UDI_CDC_SET_CODING_EXT(port,cfg)
#define  UDI_CDC_SET_DTR_EXT(port,set)
#define  UDI_CDC_SET_RTS_EXT(port,set)
/*
 * #define UDI_CDC_ENABLE_EXT(port) my_callback_cdc_enable()
 * extern bool my_callback_cdc_enable(void);
```

```
 * #define UDI_CDC_DISABLE_EXT(port) my_callback_cdc_disable()
 * extern void my_callback_cdc_disable(void);
 * #define  UDI_CDC_RX_NOTIFY(port) my_callback_rx_notify(port)
 * extern void my_callback_rx_notify(uint8_t port);
 * #define  UDI_CDC_TX_EMPTY_NOTIFY(port) my_callback_tx_empty_notify(port)
 * extern void my_callback_tx_empty_notify(uint8_t port);
 * #define  UDI_CDC_SET_CODING_EXT(port,cfg) my_callback_config(port,cfg)
 * extern void my_callback_config(uint8_t port, usb_cdc_line_coding_t * cfg);
 * #define  UDI_CDC_SET_DTR_EXT(port,set) my_callback_cdc_set_dtr(port,set)
 * extern void my_callback_cdc_set_dtr(uint8_t port, bool b_enable);
 * #define  UDI_CDC_SET_RTS_EXT(port,set) my_callback_cdc_set_rts(port,set)
 * extern void my_callback_cdc_set_rts(uint8_t port, bool b_enable);
 */

#define  UDI_CDC_LOW_RATE

#define  UDI_CDC_DEFAULT_RATE             115200
#define  UDI_CDC_DEFAULT_STOPBITS         CDC_STOP_BITS_1
#define  UDI_CDC_DEFAULT_PARITY           CDC_PAR_NONE
#define  UDI_CDC_DEFAULT_DATABITS         8

#define  UDI_CDC_DATA_EP_IN_0          (1 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_0         (2 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_0             (3 | USB_EP_DIR_IN)  // Notify endpoint
#define  UDI_CDC_DATA_EP_IN_2          (4 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_2         (5 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_2             (6 | USB_EP_DIR_IN)  // Notify endpoint
#define  UDI_CDC_DATA_EP_IN_3          (7 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_3         (8 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_3             (9 | USB_EP_DIR_IN)  // Notify endpoint

#define  UDI_CDC_COMM_IFACE_NUMBER_0   0
#define  UDI_CDC_DATA_IFACE_NUMBER_0   1
#define  UDI_CDC_COMM_IFACE_NUMBER_2   2
#define  UDI_CDC_DATA_IFACE_NUMBER_2   3
#define  UDI_CDC_COMM_IFACE_NUMBER_3   4
#define  UDI_CDC_DATA_IFACE_NUMBER_3   5



#define UDI_MSC_GLOBAL_VENDOR_ID            \
   'A', 'T', 'M', 'E', 'L', ' ', ' ', ' '
#define UDI_MSC_GLOBAL_PRODUCT_VERSION           \
   '1', '.', '0', '0'

#define  UDI_MSC_ENABLE_EXT()          true
#define  UDI_MSC_DISABLE_EXT()
#define  UDI_MSC_NOTIFY_TRANS_EXT()
/*
 * #define UDI_MSC_ENABLE_EXT() my_callback_msc_enable()
 * extern bool my_callback_msc_enable(void);
 * #define UDI_MSC_DISABLE_EXT() my_callback_msc_disable()
 * extern void my_callback_msc_disable(void);
 * #define  UDI_MSC_NOTIFY_TRANS_EXT()    msc_notify_trans()
 * extern void msc_notify_trans(void) {
 */

#define  UDI_MSC_EP_IN                 (1 | USB_EP_DIR_IN)
#define  UDI_MSC_EP_OUT                (2 | USB_EP_DIR_OUT)
```

```
#define  UDI_MSC_IFACE_NUMBER          0



#define  UDI_HID_MOUSE_ENABLE_EXT()       true
#define  UDI_HID_MOUSE_DISABLE_EXT()
// #define UDI_HID_MOUSE_ENABLE_EXT() my_callback_mouse_enable()
// extern bool my_callback_mouse_enable(void);
// #define UDI_HID_MOUSE_DISABLE_EXT() my_callback_mouse_disable()
// extern void my_callback_mouse_disable(void);

#define  UDI_HID_MOUSE_EP_IN           (1 | USB_EP_DIR_IN)

#define  UDI_HID_MOUSE_IFACE_NUMBER    0



#define  UDI_HID_KBD_ENABLE_EXT()       true
#define  UDI_HID_KBD_DISABLE_EXT()
// #define UDI_HID_KBD_ENABLE_EXT() my_callback_keyboard_enable()
// extern bool my_callback_keyboard_enable(void);
// #define UDI_HID_KBD_DISABLE_EXT() my_callback_keyboard_disable()
// extern void my_callback_keyboard_disable(void);
#define  UDI_HID_KBD_CHANGE_LED(value)
// #define  UDI_HID_KBD_CHANGE_LED(value) my_callback_keyboard_led(value)
// extern void my_callback_keyboard_led(uint8_t value)

#define  UDI_HID_KBD_EP_IN           (1 | USB_EP_DIR_IN)

#define  UDI_HID_KBD_IFACE_NUMBER    0



#define  UDI_HID_GENERIC_ENABLE_EXT()       true
#define  UDI_HID_GENERIC_DISABLE_EXT()
#define  UDI_HID_GENERIC_REPORT_OUT(ptr)
#define  UDI_HID_GENERIC_SET_FEATURE(f)
/*
 * #define UDI_HID_GENERIC_ENABLE_EXT() my_callback_generic_enable()
 * extern bool my_callback_generic_enable(void);
 * #define UDI_HID_GENERIC_DISABLE_EXT() my_callback_generic_disable()
 * extern void my_callback_generic_disable(void);
 * #define  UDI_HID_GENERIC_REPORT_OUT(ptr) my_callback_generic_report_out(ptr)
 * extern void my_callback_generic_report_out(uint8_t *report);
 * #define  UDI_HID_GENERIC_SET_FEATURE(f) my_callback_generic_set_feature(f)
 * extern void my_callback_generic_set_feature(uint8_t *report_feature);
 */
#define  UDI_HID_REPORT_IN_SIZE             64
#define  UDI_HID_REPORT_OUT_SIZE            64
#define  UDI_HID_REPORT_FEATURE_SIZE        4
#define  UDI_HID_GENERIC_EP_SIZE            64

#define  UDI_HID_GENERIC_EP_OUT   (2 | USB_EP_DIR_OUT)
#define  UDI_HID_GENERIC_EP_IN    (1 | USB_EP_DIR_IN)

#define  UDI_HID_GENERIC_IFACE_NUMBER    0
```

```
#define  UDI_PHDC_ENABLE_EXT()          true
#define  UDI_PHDC_DISABLE_EXT()

#define  UDI_PHDC_DATAMSG_FORMAT       USB_PHDC_DATAMSG_FORMAT_11073_20601
#define  UDI_PHDC_SPECIALIZATION       {0x2345} // Define in 11073_20601

#define  UDI_PHDC_QOS_OUT              \
        (USB_PHDC_QOS_MEDIUM_BETTER|USB_PHDC_QOS_HIGH_BEST)
#define  UDI_PHDC_QOS_IN               \
        (USB_PHDC_QOS_LOW_GOOD|USB_PHDC_QOS_MEDIUM_BETTER|USB_PHDC_QOS_MEDIUM_BEST)

#define  UDI_PHDC_METADATA_DESC_BULK_IN   {0x01,0x02,0x03}
#define  UDI_PHDC_METADATA_DESC_BULK_OUT  {0x01,0x02,0x03}
#define  UDI_PHDC_METADATA_DESC_INT_IN    {0x01,0x02,0x03}

#define  UDI_PHDC_EP_BULK_OUT          (1 | USB_EP_DIR_OUT)
#define  UDI_PHDC_EP_BULK_IN           (2 | USB_EP_DIR_IN)
#if ((UDI_PHDC_QOS_IN&USB_PHDC_QOS_LOW_GOOD)==USB_PHDC_QOS_LOW_GOOD)
// Only if UDI_PHDC_QOS_IN include USB_PHDC_QOS_LOW_GOOD
#  define  UDI_PHDC_EP_INTERRUPT_IN      (3 | USB_EP_DIR_IN)
#endif

#define  UDI_PHDC_EP_SIZE_BULK_OUT     32
#define  UDI_PHDC_EP_SIZE_BULK_IN      32
#define  UDI_PHDC_EP_SIZE_INT_IN       8


#define  UDI_PHDC_IFACE_NUMBER          0



#define  UDI_VENDOR_ENABLE_EXT()          true
#define  UDI_VENDOR_DISABLE_EXT()
#define  UDI_VENDOR_SETUP_OUT_RECEIVED()  false
#define  UDI_VENDOR_SETUP_IN_RECEIVED()   false
/*
 * #define UDI_VENDOR_ENABLE_EXT() my_callback_vendor_enable()
 * extern bool my_callback_vendor_enable(void);
 * #define UDI_VENDOR_DISABLE_EXT() my_callback_vendor_disable()
 * extern void my_callback_vendor_disable(void);
 *
 * #define  UDI_VENDOR_SETUP_OUT_RECEIVED()  my_vendor_setup_out_received()
 * extern bool my_vendor_setup_out_received(void);
 * #define  UDI_VENDOR_SETUP_IN_RECEIVED()   my_vendor_setup_in_received()
 * extern bool my_vendor_setup_in_received(void);
 */

#define  UDI_VENDOR_EPS_SIZE_INT_FS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_FS   64
#define  UDI_VENDOR_EPS_SIZE_ISO_FS   256

#define  UDI_VENDOR_EPS_SIZE_INT_HS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_HS  512
#define  UDI_VENDOR_EPS_SIZE_ISO_HS    64

#define  UDI_VENDOR_EP_INTERRUPT_IN  (1 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_INTERRUPT_OUT (2 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_BULK_IN       (3 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_BULK_OUT      (4 | USB_EP_DIR_OUT)
```

```
#define  UDI_VENDOR_EP_ISO_IN       (5 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_ISO_OUT      (6 | USB_EP_DIR_OUT)


#define  UDI_VENDOR_IFACE_NUMBER    0



//... Eventually add other Interface Configuration



#define UDI_COMPOSITE_DESC_T

#define UDI_COMPOSITE_DESC_FS

#define UDI_COMPOSITE_DESC_HS

#define UDI_COMPOSITE_API

/* Example for device with cdc, msc and hid mouse interface
#define UDI_COMPOSITE_DESC_T \
    usb_iad_desc_t udi_cdc_iad; \
    udi_cdc_comm_desc_t udi_cdc_comm; \
    udi_cdc_data_desc_t udi_cdc_data; \
    udi_msc_desc_t udi_msc; \
    udi_hid_mouse_desc_t udi_hid_mouse

#define UDI_COMPOSITE_DESC_FS \
    .udi_cdc_iad               = UDI_CDC_IAD_DESC_0, \
    .udi_cdc_comm              = UDI_CDC_COMM_DESC_0, \
    .udi_cdc_data              = UDI_CDC_DATA_DESC_0_FS, \
    .udi_msc                   = UDI_MSC_DESC_FS, \
    .udi_hid_mouse             = UDI_HID_MOUSE_DESC

#define UDI_COMPOSITE_DESC_HS \
    .udi_cdc_iad               = UDI_CDC_IAD_DESC_0, \
    .udi_cdc_comm              = UDI_CDC_COMM_DESC_0, \
    .udi_cdc_data              = UDI_CDC_DATA_DESC_0_HS, \
    .udi_msc                   = UDI_MSC_DESC_HS, \
    .udi_hid_mouse             = UDI_HID_MOUSE_DESC

#define UDI_COMPOSITE_API  \
    &udi_api_cdc_comm,       \
    &udi_api_cdc_data,       \
    &udi_api_msc,            \
    &udi_api_hid_mouse
*/




/* Example of include for interface
#include "udi_msc.h"
#include "udi_hid_kbd.h"
#include "udi_hid_mouse.h"
#include "udi_cdc.h"
#include "udi_phdc.h"
#include "udi_vendor.h"
*/
```

```
/* Declaration of callbacks used by USB
#include "callback_def.h"
*/

#endif // _CONF_USB_H_
```

4.3.2 conf_clock.h

4.3.2.1 AT32UC3A0, AT32UC3A1, AT32UC3B Devices (USBB)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL0

// ===== PLL0 Options
#define CONFIG_PLL0_SOURCE            PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
#define CONFIG_PLL0_MUL               8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL0_DIV               2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL1_MUL             8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL1_DIV             2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
//#define CONFIG_SYSCLK_CPU_DIV       0 /* Fcpu = Fsys/(2 ^ CPU_div) */
//#define CONFIG_SYSCLK_PBA_DIV       0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV       0 /* Fpbb = Fsys/(2 ^ PBB_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

4.3.2.2 AT32UC3C, ATUCXXD, ATUCXXL3U, ATUCXXL4U Devices (USBC)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC0
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_PLL0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL1
```

```
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RC8M

// ===== PLL0 Options
#define CONFIG_PLL0_SOURCE           PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE         PLL_SRC_RC8M
#define CONFIG_PLL0_MUL              8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL0_DIV              2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
//#define CONFIG_PLL1_SOURCE         PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE         PLL_SRC_RC8M
//#define CONFIG_PLL1_MUL            8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL1_DIV            2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
//#define CONFIG_SYSCLK_CPU_DIV      0 /* Fcpu = Fsys/(2 ^ CPU_div) */
//#define CONFIG_SYSCLK_PBA_DIV      0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV      0 /* Fpbb = Fsys/(2 ^ PBB_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC1
#define CONFIG_USBCLK_SOURCE         USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV            1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

4.3.2.3 SAM3X, SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock (MCK) Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_BYPASS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_4M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_8M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_12M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_BYPASS
#define CONFIG_SYSCLK_SOURCE         SYSCLK_SRC_PLLACK
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_UPLLCK

// ===== System Clock (MCK) Prescaler Options   (Fmck = Fsys / (SYSCLK_PRES))
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_1
#define CONFIG_SYSCLK_PRES           SYSCLK_PRES_2
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_4
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_8
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_16
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_32
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_64
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_3
```

```
// ===== PLL0 (A) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
// Use mul and div effective values here.
#define CONFIG_PLL0_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL0_MUL             14
#define CONFIG_PLL0_DIV             1

// ===== UPLL (UTMI) Hardware fixed at 480MHz.

// ===== USB Clock Source Options   (Fusb = FpllX / USB_div)
// Use div effective value here.
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_UPLL
#define CONFIG_USBCLK_DIV           1

// ===== Target frequency (System clock)
// - XTAL frequency: 12MHz
// - System clock source: PLLA
// - System clock prescaler: 2 (divided by 2)
// - PLLA source: XTAL
// - PLLA output: XTAL * 14 / 1
// - System clock is: 12 * 14 / 1 /2 = 84MHz
// ===== Target frequency (USB Clock)
// - USB clock source: UPLL
// - USB clock divider: 1 (not divided)
// - UPLL frequency: 480MHz
// - USB clock: 480 / 1 = 480MHz


#endif /* CONF_CLOCK_H_INCLUDED */
```

4.3.3 conf_clocks.h

4.3.3.1 SAMD21 Device (USB)

```
#include <clock.h>

#ifndef CONF_CLOCKS_H_INCLUDED
#  define CONF_CLOCKS_H_INCLUDED

/* System clock bus configuration */
#  define CONF_CLOCK_CPU_CLOCK_FAILURE_DETECT     false
#  define CONF_CLOCK_FLASH_WAIT_STATES            2
#  define CONF_CLOCK_CPU_DIVIDER                  SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBA_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBB_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1

/* SYSTEM_CLOCK_SOURCE_OSC8M configuration - Internal 8MHz oscillator */
#  define CONF_CLOCK_OSC8M_PRESCALER              SYSTEM_OSC8M_DIV_1
#  define CONF_CLOCK_OSC8M_ON_DEMAND              true
#  define CONF_CLOCK_OSC8M_RUN_IN_STANDBY         false

/* SYSTEM_CLOCK_SOURCE_XOSC configuration - External clock/oscillator */
#  define CONF_CLOCK_XOSC_ENABLE                  false
#  define CONF_CLOCK_XOSC_EXTERNAL_CRYSTAL        SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC_EXTERNAL_FREQUENCY      12000000UL
#  define CONF_CLOCK_XOSC_STARTUP_TIME            SYSTEM_XOSC_STARTUP_32768
#  define CONF_CLOCK_XOSC_AUTO_GAIN_CONTROL       true
#  define CONF_CLOCK_XOSC_ON_DEMAND               true
#  define CONF_CLOCK_XOSC_RUN_IN_STANDBY          false
```

```
/* SYSTEM_CLOCK_SOURCE_XOSC32K configuration - External 32KHz crystal/clock oscillator */
#  define CONF_CLOCK_XOSC32K_ENABLE                false
#  define CONF_CLOCK_XOSC32K_EXTERNAL_CRYSTAL      SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC32K_STARTUP_TIME          SYSTEM_XOSC32K_STARTUP_65536
#  define CONF_CLOCK_XOSC32K_AUTO_AMPLITUDE_CONTROL  false
#  define CONF_CLOCK_XOSC32K_ENABLE_1KHZ_OUPUT     false
#  define CONF_CLOCK_XOSC32K_ENABLE_32KHZ_OUTPUT   true
#  define CONF_CLOCK_XOSC32K_ON_DEMAND             true
#  define CONF_CLOCK_XOSC32K_RUN_IN_STANDBY        false

/* SYSTEM_CLOCK_SOURCE_OSC32K configuration - Internal 32KHz oscillator */
#  define CONF_CLOCK_OSC32K_ENABLE                 false
#  define CONF_CLOCK_OSC32K_STARTUP_TIME           SYSTEM_OSC32K_STARTUP_130
#  define CONF_CLOCK_OSC32K_ENABLE_1KHZ_OUTPUT     true
#  define CONF_CLOCK_OSC32K_ENABLE_32KHZ_OUTPUT    true
#  define CONF_CLOCK_OSC32K_ON_DEMAND              true
#  define CONF_CLOCK_OSC32K_RUN_IN_STANDBY         false

/* SYSTEM_CLOCK_SOURCE_DFLL configuration - Digital Frequency Locked Loop */
#  define CONF_CLOCK_DFLL_ENABLE                   true
#  define CONF_CLOCK_DFLL_LOOP_MODE                SYSTEM_CLOCK_DFLL_LOOP_MODE_USB_RECOVERY
#  define CONF_CLOCK_DFLL_ON_DEMAND                true

/* DFLL open loop mode configuration */
#  define CONF_CLOCK_DFLL_COARSE_VALUE             (0x1f / 4)
#  define CONF_CLOCK_DFLL_FINE_VALUE               (0xff / 4)

/* DFLL closed loop mode configuration */
#  define CONF_CLOCK_DFLL_SOURCE_GCLK_GENERATOR    GCLK_GENERATOR_1
#  define CONF_CLOCK_DFLL_MULTIPLY_FACTOR          (48000000 / 32768)
#  define CONF_CLOCK_DFLL_QUICK_LOCK               true
#  define CONF_CLOCK_DFLL_TRACK_AFTER_FINE_LOCK    true
#  define CONF_CLOCK_DFLL_KEEP_LOCK_ON_WAKEUP      true
#  define CONF_CLOCK_DFLL_ENABLE_CHILL_CYCLE       true
#  define CONF_CLOCK_DFLL_MAX_COARSE_STEP_SIZE     (0x1f / 4)
#  define CONF_CLOCK_DFLL_MAX_FINE_STEP_SIZE       (0xff / 4)

/* SYSTEM_CLOCK_SOURCE_DPLL configuration - Digital Phase-Locked Loop */
#  define CONF_CLOCK_DPLL_ENABLE                   false
#  define CONF_CLOCK_DPLL_ON_DEMAND                true
#  define CONF_CLOCK_DPLL_RUN_IN_STANDBY           false
#  define CONF_CLOCK_DPLL_LOCK_BYPASS              false
#  define CONF_CLOCK_DPLL_WAKE_UP_FAST             false
#  define CONF_CLOCK_DPLL_LOW_POWER_ENABLE         false

#  define CONF_CLOCK_DPLL_LOCK_TIME                SYSTEM_CLOCK_SOURCE_DPLL_LOCK_TIME_NO_TIMEOUT
#  define CONF_CLOCK_DPLL_REFERENCE_CLOCK          SYSTEM_CLOCK_SOURCE_DPLL_REFERENCE_CLOCK_REF0
#  define CONF_CLOCK_DPLL_FILTER                   SYSTEM_CLOCK_SOURCE_DPLL_FILTER_DEFAULT

#  define CONF_CLOCK_DPLL_REFERENCE_FREQUENCY      32768
#  define CONF_CLOCK_DPLL_REFEREMCE_DIVIDER        1
#  define CONF_CLOCK_DPLL_OUTPUT_FREQUENCY         48000000

/* Set this to true to configure the GCLK when running clocks_init. If set to
 * false, none of the GCLK generators will be configured in clocks_init(). */
#  define CONF_CLOCK_CONFIGURE_GCLK                true

/* Configure GCLK generator 0 (Main Clock) */
#  define CONF_CLOCK_GCLK_0_ENABLE                 true
#  define CONF_CLOCK_GCLK_0_RUN_IN_STANDBY         true
```

```
#  define CONF_CLOCK_GCLK_0_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_DFLL
#  define CONF_CLOCK_GCLK_0_PRESCALER             1
#  define CONF_CLOCK_GCLK_0_OUTPUT_ENABLE         false

/* Configure GCLK generator 1 */
#  define CONF_CLOCK_GCLK_1_ENABLE                false
#  define CONF_CLOCK_GCLK_1_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_1_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_XOSC32K
#  define CONF_CLOCK_GCLK_1_PRESCALER             1
#  define CONF_CLOCK_GCLK_1_OUTPUT_ENABLE         false

/* Configure GCLK generator 2 (RTC) */
#  define CONF_CLOCK_GCLK_2_ENABLE                false
#  define CONF_CLOCK_GCLK_2_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_2_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC32K
#  define CONF_CLOCK_GCLK_2_PRESCALER             32
#  define CONF_CLOCK_GCLK_2_OUTPUT_ENABLE         false

/* Configure GCLK generator 3 */
#  define CONF_CLOCK_GCLK_3_ENABLE                false
#  define CONF_CLOCK_GCLK_3_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_3_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_3_PRESCALER             1
#  define CONF_CLOCK_GCLK_3_OUTPUT_ENABLE         false

/* Configure GCLK generator 4 */
#  define CONF_CLOCK_GCLK_4_ENABLE                false
#  define CONF_CLOCK_GCLK_4_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_4_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_4_PRESCALER             1
#  define CONF_CLOCK_GCLK_4_OUTPUT_ENABLE         false

/* Configure GCLK generator 5 */
#  define CONF_CLOCK_GCLK_5_ENABLE                false
#  define CONF_CLOCK_GCLK_5_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_5_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_5_PRESCALER             1
#  define CONF_CLOCK_GCLK_5_OUTPUT_ENABLE         false

/* Configure GCLK generator 6 */
#  define CONF_CLOCK_GCLK_6_ENABLE                false
#  define CONF_CLOCK_GCLK_6_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_6_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_6_PRESCALER             1
#  define CONF_CLOCK_GCLK_6_OUTPUT_ENABLE         false

/* Configure GCLK generator 7 */
#  define CONF_CLOCK_GCLK_7_ENABLE                false
#  define CONF_CLOCK_GCLK_7_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_7_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_7_PRESCALER             1
#  define CONF_CLOCK_GCLK_7_OUTPUT_ENABLE         false

#endif /* CONF_CLOCKS_H_INCLUDED */
```

4.3.4 conf_board.h

4.3.4.1 AT32UC3A0, AT32UC3A1, AT32UC3B Devices (USBB)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

4.3.4.2 AT32UC3C, ATUCXXD, ATUCXXL3U, ATUCXXL4U Devices (USBC)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

// Enable USB Port
#define CONF_BOARD_USB_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

4.3.4.3 SAM3X, SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// USB pins are used
#define CONF_BOARD_USB_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

4.3.4.4 SAMD21 Device (USB)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

/* Enable USB VBUS detect */
#define CONF_BOARD_USB_VBUS_DETECT

#endif /* CONF_BOARD_H_INCLUDED */
```

5. USB Device Interface (UDI) for Human Interface Device Mouse (HID Mouse)

USB Device Interface (UDI) for Human Interface Device Mouse (HID mouse) provides an interface for the configuration and management of USB HID mouse device.

The outline of this documentation is as follows:

- API Overview
- Quick Start Guide for USB Device Mouse Module (UDI Mouse)
- Configuration File Examples

For more details for Atmel® Software Framework (ASF) USB Device Stack and USB Device HID Mouse, refer to following application notes:

- AVR4900: ASF - USB Device Stack[1]
- AVR4903: ASF - USB Device HID Mouse Application[2]
- AVR4920: ASF - USB Device Stack - Compliance and Performance Figures[3]
- AVR4921: ASF - USB Device Stack Differences between ASF V1 and V2[4]

5.1 API Overview

5.1.1 Variable and Type Definitions

5.1.1.1 Interface with USB Device Core (UDC)

Structure required by UDC.

Variable udi_api_hid_mouse

```
UDC_DESC_STORAGE udi_api_t udi_api_hid_mouse
```

Global structure which contains standard UDI API for UDC.

5.1.2 Structure Definitions

5.1.2.1 Struct udi_hid_mouse_desc_t

Interface descriptor structure for HID mouse.

Table 5-1. Members

Type	Name	Description
usb_ep_desc_t	ep	Standard USB endpoint descriptor structure
usb_hid_descriptor_t	hid	HID Descriptor
usb_iface_desc_t	iface	Standard USB interface descriptor structure

[1] http://www.atmel.com/dyn/resources/prod_documents/doc8360.pdf
[2] http://www.atmel.com/dyn/resources/prod_documents/doc8409.pdf
[3] http://www.atmel.com/dyn/resources/prod_documents/doc8410.pdf
[4] http://www.atmel.com/dyn/resources/prod_documents/doc8411.pdf

5.1.2.2 Struct udi_hid_mouse_report_desc_t

Report descriptor for HID mouse.

Table 5-2. Members

Type	Name	Description
uint8_t	array[]	Array to put detailed report data

5.1.3 Macro Definitions

5.1.3.1 USB Interface Descriptors

The following structures provide predefined USB interface descriptors. It must be used to define the final USB descriptors.

Macro UDI_HID_MOUSE_STRING_ID

```
#define UDI_HID_MOUSE_STRING_ID 0
```

By default no string associated to this interface.

Macro UDI_HID_MOUSE_EP_SIZE

```
#define UDI_HID_MOUSE_EP_SIZE 8
```

HID mouse endpoints size.

Macro UDI_HID_MOUSE_DESC

```
#define UDI_HID_MOUSE_DESC \
  {\
 .iface.bLength             = sizeof(usb_iface_desc_t),\
 .iface.bDescriptorType     = USB_DT_INTERFACE,\
 .iface.bInterfaceNumber    = UDI_HID_MOUSE_IFACE_NUMBER,\
 .iface.bAlternateSetting   = 0,\
 .iface.bNumEndpoints       = 1,\
 .iface.bInterfaceClass     = HID_CLASS,\
 .iface.bInterfaceSubClass  = HID_SUB_CLASS_BOOT,\
 .iface.bInterfaceProtocol  = HID_PROTOCOL_MOUSE,\
 .iface.iInterface          = UDI_HID_MOUSE_STRING_ID,\
 .hid.bLength               = sizeof(usb_hid_descriptor_t),\
 .hid.bDescriptorType       = USB_DT_HID,\
 .hid.bcdHID                = LE16(USB_HID_BDC_V1_11),\
 .hid.bCountryCode          = USB_HID_NO_COUNTRY_CODE,\
 .hid.bNumDescriptors       = USB_HID_NUM_DESC,\
 .hid.bRDescriptorType      = USB_DT_HID_REPORT,\
 .hid.wDescriptorLength     = LE16(sizeof(udi_hid_mouse_report_desc_t)),\
 .ep.bLength                = sizeof(usb_ep_desc_t),\
 .ep.bDescriptorType        = USB_DT_ENDPOINT,\
 .ep.bEndpointAddress       = UDI_HID_MOUSE_EP_IN,\
 .ep.bmAttributes           = USB_EP_TYPE_INTERRUPT,\
```

```
  .ep.wMaxPacketSize          = LE16(UDI_HID_MOUSE_EP_SIZE),\
  .ep.bInterval               = 10,\
  }
```

Content of HID mouse interface descriptor for all speed.

5.1.3.2 Interfaces for Buttons Events

Macro HID_MOUSE_BTN_DOWN

```
#define HID_MOUSE_BTN_DOWN true
```

Value to signal a button down (pressed).

Macro HID_MOUSE_BTN_UP

```
#define HID_MOUSE_BTN_UP false
```

Value to signal a button up (released).

5.1.4 Function Definitions

5.1.4.1 Interfaces for Mouse Events

Function udi_hid_mouse_moveScroll()

Move the scroll wheel.

```
bool udi_hid_mouse_moveScroll(
  int8_t pos)
```

Table 5-3. Parameters

Data direction	Parameter name	Description
[in]	pos	Signed value to move

Returns

1 if function was successfully done, otherwise 0.

Function udi_hid_mouse_moveY()

Move the mouse pointer on Y axe.

```
bool udi_hid_mouse_moveY(
  int8_t pos_y)
```

Table 5-4. Parameters

Data direction	Parameter name	Description
[in]	pos_y	Signed value to move

Returns 1 if function was successfully done, otherwise 0.

Function udi_hid_mouse_moveX()

Move the mouse pointer on X axe.

```
bool udi_hid_mouse_moveX(
  int8_t pos_x)
```

Table 5-5. Parameters

Data direction	Parameter name	Description
[in]	pos_x	Signed value to move

Returns 1 if function was successfully done, otherwise 0.

5.1.4.2 Interfaces for Buttons Events

Function udi_hid_mouse_btnmiddle()

Changes middle button state.

```
bool udi_hid_mouse_btnmiddle(
  bool b_state)
```

Table 5-6. Parameters

Data direction	Parameter name	Description
[in]	b_state	New button state

Returns 1 if function was successfully done, otherwise 0.

Function udi_hid_mouse_btnright()

Changes right button state.

```
bool udi_hid_mouse_btnright(
  bool b_state)
```

Table 5-7. Parameters

Data direction	Parameter name	Description
[in]	b_state	New button state

Returns 1 if function was successfully done, otherwise 0.

Function udi_hid_mouse_btnleft()

Changes left button state.

```
bool udi_hid_mouse_btnleft(
  bool b_state)
```

Table 5-8. Parameters

Data direction	Parameter name	Description
[in]	b_state	New button state

Returns 1 if function was successfully done, otherwise 0.

5.2 Quick Start Guide for USB Device Mouse Module (UDI Mouse)

This is the quick start guide for the USB Device Mouse Module (UDI Mouse) with step-by-step instructions on how to configure and use the modules in a selection of use cases.

The use cases contain several code fragments. The code fragments in the steps for setup can be copied into a custom initialization function, while the steps for usage can be copied into, e.g., the main application function.

5.2.1 Basic Use Case

In this basic use case, the "USB HID Mouse (Single Interface Device)" module is used. The "USB HID Mouse (Composite Device)" module usage is described in Advanced Use Cases.

5.2.2 Setup Steps

As a USB device, it follows common USB device setup steps. Refer to USB Device Basic Setup.

5.2.3 Usage Steps

5.2.3.1 Example Code

Content of conf_usb.h:

```
#define UDI_HID_MOUSE_ENABLE_EXT() my_callback_mouse_enable()
extern bool my_callback_mouse_enable(void);
#define UDI_HID_MOUSE_DISABLE_EXT() my_callback_mouse_disable()
extern void my_callback_mouse_disable(void);
#include "udi_hid_mouse_conf.h" // At the end of conf_usb.h file
```

Add to application C-file:

```
static bool my_flag_autorize_mouse_events = false;
bool my_callback_mouse_enable(void)
{
   my_flag_autorize_mouse_events = true;
   return true;
}
void my_callback_mouse_disable(void)
{
   my_flag_autorize_mouse_events = false;
}
```

```
void my_button_press_event(void)
{
   if (!my_flag_autorize_mouse_events) {
      return;
   }
   udi_hid_mouse_btnleft(HID_MOUSE_BTN_DOWN);
}
```

5.2.3.2 Workflow

1. Ensure that conf_usb.h is available and contains the following configuration which is the USB device mouse configuration:

```
#define UDI_HID_MOUSE_ENABLE_EXT() my_callback_mouse_enable()
extern bool my_callback_mouse_enable(void);
```

Note

After the device enumeration (detecting and identifying USB devices), the USB host starts the device configuration. When the USB mouse interface from the device is accepted by the host, the USB host enables this interface and the UDI_HID_MOUSE_ENABLE_EXT() callback function is called and return true. Thus, it is recommended to enable sensors used by the mouse in this function.

```
#define UDI_HID_MOUSE_DISABLE_EXT() my_callback_mouse_disable()
extern void my_callback_mouse_disable(void);
```

Note

When the USB device is unplugged or is reset by the USB host, the USB interface is disabled and the UDI_HID_MOUSE_DISABLE_EXT() callback function is called. Thus, it is recommended to disable sensors used by the mouse in this function.

2. Send mouse events:

```
// Sends a value at scroll wheel
udi_hid_mouse_moveScroll(int8_t pos);
// Sends an Y axis value at mouse pointer
udi_hid_mouse_moveY(int8_t pos_y);
// Sends an X axis value at mouse pointer
udi_hid_mouse_moveX(int8_t pos_x);
// Sends a middle click event
udi_hid_mouse_btnmiddle(bool b_state);
// Sends a right click event
udi_hid_mouse_btnright(bool b_state);
// Sends a left click event
udi_hid_mouse_btnleft(bool b_state);
```

5.2.4 Advanced Use Cases

For multiple interface use of UDI HID module, see the following:

- HID Mouse in a Composite Device

For more advanced use of the UDI HID mouse module, see the following:

- USB Device Advanced Use Cases

5.2.5 HID Mouse in a Composite Device

A USB Composite Device is a USB Device which uses more than one USB class. In this use case, the "USB HID Mouse (Composite Device)" module is used to create a USB composite device. Thus, this USB module can be associated with another "Composite Device" module, like "USB MSC (Composite Device)".

Also, you can refer to application note AVR4902 ASF - USB Composite Device[5].

5.2.5.1 Setup Steps

For the setup code of this use case to work, the Basic Use Case must be followed.

5.2.5.2 Usage Steps

Example Code

Content of conf_usb.h:

```
#define USB_DEVICE_EP_CTRL_SIZE  64
#define USB_DEVICE_NB_INTERFACE (X+1)
#define USB_DEVICE_MAX_EP (X+1)

#define UDI_HID_MOUSE_EP_IN  (X | USB_EP_DIR_IN)
#define UDI_HID_MOUSE_IFACE_NUMBER  X

#define UDI_COMPOSITE_DESC_T \
   udi_hid_mouse_desc_t udi_hid_mouse; \
   ...
#define UDI_COMPOSITE_DESC_FS \
   .udi_hid_mouse = UDI_HID_MOUSE_DESC, \
   ...
#define UDI_COMPOSITE_DESC_HS \
   .udi_hid_mouse = UDI_HID_MOUSE_DESC, \
   ...
#define UDI_COMPOSITE_API \
   &udi_api_hid_mouse, \
   ...
```

Workflow

1. Ensure that conf_usb.h is available and contains the following parameters required for a USB composite device configuration:

```
// Endpoint control size, This must be:
// - 8 for low speed
// - 8, 16, 32 or 64 for full speed device (8 is recommended to save RAM)
// - 64 for a high speed device
#define USB_DEVICE_EP_CTRL_SIZE  64
// Total Number of interfaces on this USB device.
// Add 1 for HID mouse.
#define USB_DEVICE_NB_INTERFACE (X+1)
// Total number of endpoints on this USB device.
// This must include each endpoint for each interface.
// Add 1 for HID mouse.
#define USB_DEVICE_MAX_EP (X+1)
```

2. Ensure that conf_usb.h contains the description of composite device:

```
// The endpoint number chosen by you for the mouse.
```

[5] http://www.atmel.com/dyn/resources/prod_documents/doc8445.pdf

```
// The endpoint number starting from 1.
#define UDI_HID_MOUSE_EP_IN  (X | USB_EP_DIR_IN)
// The interface index of an interface starting from 0
#define UDI_HID_MOUSE_IFACE_NUMBER  X
```

3. Ensure that conf_usb.h contains the following parameters required for a USB composite device configuration:

```
// USB Interfaces descriptor structure
#define UDI_COMPOSITE_DESC_T \
   ...
   udi_hid_mouse_desc_t udi_hid_mouse; \
   ...
// USB Interfaces descriptor value for Full Speed
#define UDI_COMPOSITE_DESC_FS \
   ...
   .udi_hid_mouse = UDI_HID_MOUSE_DESC, \
   ...
// USB Interfaces descriptor value for High Speed
#define UDI_COMPOSITE_DESC_HS \
   ...
   .udi_hid_mouse = UDI_HID_MOUSE_DESC, \
   ...
// USB Interface APIs
#define UDI_COMPOSITE_API \
   ...
   &udi_api_hid_mouse, \
   ...
```

Note The descriptors order given in the four lists above must be the same as the order defined by all interface indexes. The interface index orders are defined through UDI_X_IFACE_NUMBER defines.

5.3 Configuration File Examples

5.3.1 conf_usb.h

5.3.1.1 UDI HID MOUSE Single

```
#ifndef _CONF_USB_H_
#define _CONF_USB_H_

#include "compiler.h"

#warning You must refill the following definitions with a correct values

#define  USB_DEVICE_VENDOR_ID             USB_VID_ATMEL
#define  USB_DEVICE_PRODUCT_ID            USB_PID_ATMEL_ASF_HIDGENERIC
#define  USB_DEVICE_MAJOR_VERSION         1
#define  USB_DEVICE_MINOR_VERSION         0
#define  USB_DEVICE_POWER                 100 // Consumption on Vbus line (mA)
#define  USB_DEVICE_ATTR                  \
    (USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_BUS_POWERED)
//  (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_SELF_POWERED)
//  (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_BUS_POWERED)

// #define  USB_DEVICE_MANUFACTURE_NAME      "Manufacture name"
```

```
// #define  USB_DEVICE_PRODUCT_NAME          "Product name"
// #define  USB_DEVICE_SERIAL_NAME           "12...EF"

//#define  USB_DEVICE_LOW_SPEED

#if (UC3A3||UC3A4)
//#define  USB_DEVICE_HS_SUPPORT
#endif


// #define  UDC_VBUS_EVENT(b_vbus_high)      user_callback_vbus_action(b_vbus_high)
// extern void user_callback_vbus_action(bool b_vbus_high);
// #define  UDC_SOF_EVENT()                  user_callback_sof_action()
// extern void user_callback_sof_action(void);
// #define  UDC_SUSPEND_EVENT()              user_callback_suspend_action()
// extern void user_callback_suspend_action(void);
// #define  UDC_RESUME_EVENT()               user_callback_resume_action()
// extern void user_callback_resume_action(void);
// #define  UDC_REMOTEWAKEUP_ENABLE()        user_callback_remotewakeup_enable()
// extern void user_callback_remotewakeup_enable(void);
// #define  UDC_REMOTEWAKEUP_DISABLE()       user_callback_remotewakeup_disable()
// extern void user_callback_remotewakeup_disable(void);
// #define  UDC_GET_EXTRA_STRING()



#define  UDI_HID_GENERIC_ENABLE_EXT()       true
#define  UDI_HID_GENERIC_DISABLE_EXT()
#define  UDI_HID_GENERIC_REPORT_OUT(ptr)
#define  UDI_HID_GENERIC_SET_FEATURE(f)
/*
 * #define UDI_HID_GENERIC_ENABLE_EXT() my_callback_generic_enable()
 * extern bool my_callback_generic_enable(void);
 * #define UDI_HID_GENERIC_DISABLE_EXT() my_callback_generic_disable()
 * extern void my_callback_generic_disable(void);
 * #define  UDI_HID_GENERIC_REPORT_OUT(ptr) my_callback_generic_report_out(ptr)
 * extern void my_callback_generic_report_out(uint8_t *report);
 * #define  UDI_HID_GENERIC_SET_FEATURE(f) my_callback_generic_set_feature(f)
 * extern void my_callback_generic_set_feature(uint8_t *report_feature);
 */

#define  UDI_HID_REPORT_IN_SIZE             64
#define  UDI_HID_REPORT_OUT_SIZE            64
#define  UDI_HID_REPORT_FEATURE_SIZE        4

#define  UDI_HID_GENERIC_EP_SIZE            64




#include "udi_hid_generic_conf.h"

#endif // _CONF_USB_H_
```

5.3.1.2 UDI HID MOUSE Multiple (Composite)

```
#ifndef _CONF_USB_H_
#define _CONF_USB_H_
```

```
#include "compiler.h"

#warning You must refill the following definitions with a correct values

#define  USB_DEVICE_VENDOR_ID             USB_VID_ATMEL
#define  USB_DEVICE_PRODUCT_ID            0xFFFF
#define  USB_DEVICE_MAJOR_VERSION         1
#define  USB_DEVICE_MINOR_VERSION         0
#define  USB_DEVICE_POWER                 100 // Consumption on VBUS line (mA)
#define  USB_DEVICE_ATTR                  \
        (USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_BUS_POWERED)
// (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_BUS_POWERED)

// #define  USB_DEVICE_MANUFACTURE_NAME      "Manufacture name"
// #define  USB_DEVICE_PRODUCT_NAME          "Product name"
// #define  USB_DEVICE_SERIAL_NAME           "12...EF" // Disk SN for MSC

//#define  USB_DEVICE_LOW_SPEED

#if (UC3A3||UC3A4)
//#define  USB_DEVICE_HS_SUPPORT
#endif


// #define  UDC_VBUS_EVENT(b_vbus_high)      user_callback_vbus_action(b_vbus_high)
// extern void user_callback_vbus_action(bool b_vbus_high);
// #define  UDC_SOF_EVENT()                  user_callback_sof_action()
// extern void user_callback_sof_action(void);
// #define  UDC_SUSPEND_EVENT()              user_callback_suspend_action()
// extern void user_callback_suspend_action(void);
// #define  UDC_RESUME_EVENT()               user_callback_resume_action()
// extern void user_callback_resume_action(void);
// #define  UDC_REMOTEWAKEUP_ENABLE()        user_callback_remotewakeup_enable()
// extern void user_callback_remotewakeup_enable(void);
// #define  UDC_REMOTEWAKEUP_DISABLE()       user_callback_remotewakeup_disable()
// extern void user_callback_remotewakeup_disable(void);
// #define  UDC_GET_EXTRA_STRING()

#define  USB_DEVICE_EP_CTRL_SIZE       64

#define  USB_DEVICE_NB_INTERFACE       1 // 1 or more

#define  USB_DEVICE_MAX_EP             1 // 0 to max endpoint requested by interfaces




#define  UDI_CDC_PORT_NB 1

#define  UDI_CDC_ENABLE_EXT(port)             true
#define  UDI_CDC_DISABLE_EXT(port)
#define  UDI_CDC_RX_NOTIFY(port)
#define  UDI_CDC_TX_EMPTY_NOTIFY(port)
#define  UDI_CDC_SET_CODING_EXT(port,cfg)
#define  UDI_CDC_SET_DTR_EXT(port,set)
#define  UDI_CDC_SET_RTS_EXT(port,set)
/*
 * #define UDI_CDC_ENABLE_EXT(port) my_callback_cdc_enable()
 * extern bool my_callback_cdc_enable(void);
```

```
 * #define UDI_CDC_DISABLE_EXT(port) my_callback_cdc_disable()
 * extern void my_callback_cdc_disable(void);
 * #define  UDI_CDC_RX_NOTIFY(port) my_callback_rx_notify(port)
 * extern void my_callback_rx_notify(uint8_t port);
 * #define  UDI_CDC_TX_EMPTY_NOTIFY(port) my_callback_tx_empty_notify(port)
 * extern void my_callback_tx_empty_notify(uint8_t port);
 * #define  UDI_CDC_SET_CODING_EXT(port,cfg) my_callback_config(port,cfg)
 * extern void my_callback_config(uint8_t port, usb_cdc_line_coding_t * cfg);
 * #define  UDI_CDC_SET_DTR_EXT(port,set) my_callback_cdc_set_dtr(port,set)
 * extern void my_callback_cdc_set_dtr(uint8_t port, bool b_enable);
 * #define  UDI_CDC_SET_RTS_EXT(port,set) my_callback_cdc_set_rts(port,set)
 * extern void my_callback_cdc_set_rts(uint8_t port, bool b_enable);
 */

#define  UDI_CDC_LOW_RATE

#define  UDI_CDC_DEFAULT_RATE             115200
#define  UDI_CDC_DEFAULT_STOPBITS         CDC_STOP_BITS_1
#define  UDI_CDC_DEFAULT_PARITY           CDC_PAR_NONE
#define  UDI_CDC_DEFAULT_DATABITS         8

#define  UDI_CDC_DATA_EP_IN_0          (1 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_0         (2 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_0             (3 | USB_EP_DIR_IN)  // Notify endpoint
#define  UDI_CDC_DATA_EP_IN_2          (4 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_2         (5 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_2             (6 | USB_EP_DIR_IN)  // Notify endpoint
#define  UDI_CDC_DATA_EP_IN_3          (7 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_3         (8 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_3             (9 | USB_EP_DIR_IN)  // Notify endpoint

#define  UDI_CDC_COMM_IFACE_NUMBER_0   0
#define  UDI_CDC_DATA_IFACE_NUMBER_0   1
#define  UDI_CDC_COMM_IFACE_NUMBER_2   2
#define  UDI_CDC_DATA_IFACE_NUMBER_2   3
#define  UDI_CDC_COMM_IFACE_NUMBER_3   4
#define  UDI_CDC_DATA_IFACE_NUMBER_3   5



#define UDI_MSC_GLOBAL_VENDOR_ID            \
   'A', 'T', 'M', 'E', 'L', ' ', ' ', ' '
#define UDI_MSC_GLOBAL_PRODUCT_VERSION           \
   '1', '.', '0', '0'

#define  UDI_MSC_ENABLE_EXT()          true
#define  UDI_MSC_DISABLE_EXT()
#define  UDI_MSC_NOTIFY_TRANS_EXT()
/*
 * #define UDI_MSC_ENABLE_EXT() my_callback_msc_enable()
 * extern bool my_callback_msc_enable(void);
 * #define UDI_MSC_DISABLE_EXT() my_callback_msc_disable()
 * extern void my_callback_msc_disable(void);
 * #define  UDI_MSC_NOTIFY_TRANS_EXT()    msc_notify_trans()
 * extern void msc_notify_trans(void) {
 */

#define  UDI_MSC_EP_IN                 (1 | USB_EP_DIR_IN)
#define  UDI_MSC_EP_OUT                (2 | USB_EP_DIR_OUT)
```

```
#define  UDI_MSC_IFACE_NUMBER          0



#define  UDI_HID_MOUSE_ENABLE_EXT()       true
#define  UDI_HID_MOUSE_DISABLE_EXT()
// #define UDI_HID_MOUSE_ENABLE_EXT() my_callback_mouse_enable()
// extern bool my_callback_mouse_enable(void);
// #define UDI_HID_MOUSE_DISABLE_EXT() my_callback_mouse_disable()
// extern void my_callback_mouse_disable(void);

#define  UDI_HID_MOUSE_EP_IN           (1 | USB_EP_DIR_IN)

#define  UDI_HID_MOUSE_IFACE_NUMBER    0



#define  UDI_HID_KBD_ENABLE_EXT()       true
#define  UDI_HID_KBD_DISABLE_EXT()
// #define UDI_HID_KBD_ENABLE_EXT() my_callback_keyboard_enable()
// extern bool my_callback_keyboard_enable(void);
// #define UDI_HID_KBD_DISABLE_EXT() my_callback_keyboard_disable()
// extern void my_callback_keyboard_disable(void);
#define  UDI_HID_KBD_CHANGE_LED(value)
// #define  UDI_HID_KBD_CHANGE_LED(value) my_callback_keyboard_led(value)
// extern void my_callback_keyboard_led(uint8_t value)

#define  UDI_HID_KBD_EP_IN           (1 | USB_EP_DIR_IN)

#define  UDI_HID_KBD_IFACE_NUMBER    0



#define  UDI_HID_GENERIC_ENABLE_EXT()       true
#define  UDI_HID_GENERIC_DISABLE_EXT()
#define  UDI_HID_GENERIC_REPORT_OUT(ptr)
#define  UDI_HID_GENERIC_SET_FEATURE(f)
/*
 * #define UDI_HID_GENERIC_ENABLE_EXT() my_callback_generic_enable()
 * extern bool my_callback_generic_enable(void);
 * #define UDI_HID_GENERIC_DISABLE_EXT() my_callback_generic_disable()
 * extern void my_callback_generic_disable(void);
 * #define  UDI_HID_GENERIC_REPORT_OUT(ptr) my_callback_generic_report_out(ptr)
 * extern void my_callback_generic_report_out(uint8_t *report);
 * #define  UDI_HID_GENERIC_SET_FEATURE(f) my_callback_generic_set_feature(f)
 * extern void my_callback_generic_set_feature(uint8_t *report_feature);
 */
#define  UDI_HID_REPORT_IN_SIZE             64
#define  UDI_HID_REPORT_OUT_SIZE            64
#define  UDI_HID_REPORT_FEATURE_SIZE        4
#define  UDI_HID_GENERIC_EP_SIZE            64

#define  UDI_HID_GENERIC_EP_OUT   (2 | USB_EP_DIR_OUT)
#define  UDI_HID_GENERIC_EP_IN    (1 | USB_EP_DIR_IN)

#define  UDI_HID_GENERIC_IFACE_NUMBER    0
```

```
#define  UDI_PHDC_ENABLE_EXT()          true
#define  UDI_PHDC_DISABLE_EXT()

#define  UDI_PHDC_DATAMSG_FORMAT       USB_PHDC_DATAMSG_FORMAT_11073_20601
#define  UDI_PHDC_SPECIALIZATION       {0x2345} // Define in 11073_20601

#define  UDI_PHDC_QOS_OUT              \
        (USB_PHDC_QOS_MEDIUM_BETTER|USB_PHDC_QOS_HIGH_BEST)
#define  UDI_PHDC_QOS_IN               \
        (USB_PHDC_QOS_LOW_GOOD|USB_PHDC_QOS_MEDIUM_BETTER|USB_PHDC_QOS_MEDIUM_BEST)

#define  UDI_PHDC_METADATA_DESC_BULK_IN   {0x01,0x02,0x03}
#define  UDI_PHDC_METADATA_DESC_BULK_OUT  {0x01,0x02,0x03}
#define  UDI_PHDC_METADATA_DESC_INT_IN    {0x01,0x02,0x03}

#define  UDI_PHDC_EP_BULK_OUT          (1 | USB_EP_DIR_OUT)
#define  UDI_PHDC_EP_BULK_IN           (2 | USB_EP_DIR_IN)
#if ((UDI_PHDC_QOS_IN&USB_PHDC_QOS_LOW_GOOD)==USB_PHDC_QOS_LOW_GOOD)
// Only if UDI_PHDC_QOS_IN include USB_PHDC_QOS_LOW_GOOD
#  define  UDI_PHDC_EP_INTERRUPT_IN      (3 | USB_EP_DIR_IN)
#endif

#define  UDI_PHDC_EP_SIZE_BULK_OUT     32
#define  UDI_PHDC_EP_SIZE_BULK_IN      32
#define  UDI_PHDC_EP_SIZE_INT_IN       8


#define  UDI_PHDC_IFACE_NUMBER          0



#define  UDI_VENDOR_ENABLE_EXT()          true
#define  UDI_VENDOR_DISABLE_EXT()
#define  UDI_VENDOR_SETUP_OUT_RECEIVED()  false
#define  UDI_VENDOR_SETUP_IN_RECEIVED()   false
/*
 * #define UDI_VENDOR_ENABLE_EXT() my_callback_vendor_enable()
 * extern bool my_callback_vendor_enable(void);
 * #define UDI_VENDOR_DISABLE_EXT() my_callback_vendor_disable()
 * extern void my_callback_vendor_disable(void);
 *
 * #define  UDI_VENDOR_SETUP_OUT_RECEIVED()  my_vendor_setup_out_received()
 * extern bool my_vendor_setup_out_received(void);
 * #define  UDI_VENDOR_SETUP_IN_RECEIVED()   my_vendor_setup_in_received()
 * extern bool my_vendor_setup_in_received(void);
 */

#define  UDI_VENDOR_EPS_SIZE_INT_FS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_FS   64
#define  UDI_VENDOR_EPS_SIZE_ISO_FS   256

#define  UDI_VENDOR_EPS_SIZE_INT_HS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_HS  512
#define  UDI_VENDOR_EPS_SIZE_ISO_HS    64

#define  UDI_VENDOR_EP_INTERRUPT_IN  (1 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_INTERRUPT_OUT (2 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_BULK_IN       (3 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_BULK_OUT      (4 | USB_EP_DIR_OUT)
```

```
#define  UDI_VENDOR_EP_ISO_IN       (5 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_ISO_OUT      (6 | USB_EP_DIR_OUT)


#define  UDI_VENDOR_IFACE_NUMBER    0



//... Eventually add other Interface Configuration



#define UDI_COMPOSITE_DESC_T

#define UDI_COMPOSITE_DESC_FS

#define UDI_COMPOSITE_DESC_HS

#define UDI_COMPOSITE_API

/* Example for device with cdc, msc and hid mouse interface
#define UDI_COMPOSITE_DESC_T \
    usb_iad_desc_t udi_cdc_iad; \
    udi_cdc_comm_desc_t udi_cdc_comm; \
    udi_cdc_data_desc_t udi_cdc_data; \
    udi_msc_desc_t udi_msc; \
    udi_hid_mouse_desc_t udi_hid_mouse

#define UDI_COMPOSITE_DESC_FS \
    .udi_cdc_iad               = UDI_CDC_IAD_DESC_0, \
    .udi_cdc_comm              = UDI_CDC_COMM_DESC_0, \
    .udi_cdc_data              = UDI_CDC_DATA_DESC_0_FS, \
    .udi_msc                   = UDI_MSC_DESC_FS, \
    .udi_hid_mouse             = UDI_HID_MOUSE_DESC

#define UDI_COMPOSITE_DESC_HS \
    .udi_cdc_iad               = UDI_CDC_IAD_DESC_0, \
    .udi_cdc_comm              = UDI_CDC_COMM_DESC_0, \
    .udi_cdc_data              = UDI_CDC_DATA_DESC_0_HS, \
    .udi_msc                   = UDI_MSC_DESC_HS, \
    .udi_hid_mouse             = UDI_HID_MOUSE_DESC

#define UDI_COMPOSITE_API  \
    &udi_api_cdc_comm,       \
    &udi_api_cdc_data,       \
    &udi_api_msc,            \
    &udi_api_hid_mouse
*/




/* Example of include for interface
#include "udi_msc.h"
#include "udi_hid_kbd.h"
#include "udi_hid_mouse.h"
#include "udi_cdc.h"
#include "udi_phdc.h"
#include "udi_vendor.h"
*/
```

```
/* Declaration of callbacks used by USB
#include "callback_def.h"
*/

#endif // _CONF_USB_H_
```

5.3.2 conf_clock.h

5.3.2.1 AT32UC3A0, AT32UC3A1, AT32UC3B Devices (USBB)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL0

// ===== PLL0 Options
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_MUL             4 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL0_DIV             1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
#define CONFIG_PLL1_SOURCE            PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
#define CONFIG_PLL1_MUL               8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL1_DIV               2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
//#define CONFIG_SYSCLK_CPU_DIV         0 /* Fcpu = Fsys/(2 ^ CPU_div) */
//#define CONFIG_SYSCLK_PBA_DIV         0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV         0 /* Fpbb = Fsys/(2 ^ PBB_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

5.3.2.2 AT32UC3A3, AT32UC3A4 Devices (USBB with High Speed Support)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL0
```

```
// ===== PLL0 Options
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_MUL             11 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL0_DIV             2  /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL1_MUL             8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL1_DIV             2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
#define CONFIG_SYSCLK_CPU_DIV         0 /* Fcpu = Fsys/(2 ^ CPU_div) */
#define CONFIG_SYSCLK_PBA_DIV         0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV       0 /* Fpbb = Fsys/(2 ^ PBB_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

5.3.2.3 AT32UC3C, ATUCXXD, ATUCXXL3U, ATUCXXL4U Devices (USBC)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC1
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_PLL0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL1
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RC8M

// ===== PLL0 Options
#define CONFIG_PLL0_SOURCE            PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_SOURCE          PLL_SRC_RC8M
#define CONFIG_PLL0_MUL               3 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL0_DIV               1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL1_SOURCE          PLL_SRC_RC8M
//#define CONFIG_PLL1_MUL             3 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL1_DIV             1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
//#define CONFIG_SYSCLK_CPU_DIV       0 /* Fcpu = Fsys/(2 ^ CPU_div) */
```

```
//#define CONFIG_SYSCLK_PBA_DIV       0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV       0 /* Fpbb = Fsys/(2 ^ PBB_div) */
//#define CONFIG_SYSCLK_PBC_DIV       0 /* Fpbc = Fsys/(2 ^ PBC_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC1
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

5.3.2.4 SAM3X, SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock (MCK) Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_BYPASS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_4M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_8M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_12M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_BYPASS
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLLACK
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_UPLLCK

// ===== System Clock (MCK) Prescaler Options   (Fmck = Fsys / (SYSCLK_PRES))
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_1
#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_2
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_4
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_8
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_16
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_32
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_64
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_3

// ===== PLL0 (A) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
// Use mul and div effective values here.
#define CONFIG_PLL0_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL0_MUL             14
#define CONFIG_PLL0_DIV             1

// ===== UPLL (UTMI) Hardware fixed at 480MHz.

// ===== USB Clock Source Options   (Fusb = FpllX / USB_div)
// Use div effective value here.
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_UPLL
#define CONFIG_USBCLK_DIV           1
```

```
// ===== Target frequency (System clock)
// - XTAL frequency: 12MHz
// - System clock source: PLLA
// - System clock prescaler: 2 (divided by 2)
// - PLLA source: XTAL
// - PLLA output: XTAL * 14 / 1
// - System clock is: 12 * 14 / 1 /2 = 84MHz
// ===== Target frequency (USB Clock)
// - USB clock source: UPLL
// - USB clock divider: 1 (not divided)
// - UPLL frequency: 480MHz
// - USB clock: 480 / 1 = 480MHz


#endif /* CONF_CLOCK_H_INCLUDED */
```

5.3.3 conf_clocks.h

5.3.3.1 SAMD21 Device (USB)

```
#include <clock.h>

#ifndef CONF_CLOCKS_H_INCLUDED
#  define CONF_CLOCKS_H_INCLUDED

/* System clock bus configuration */
#  define CONF_CLOCK_CPU_CLOCK_FAILURE_DETECT     false
#  define CONF_CLOCK_FLASH_WAIT_STATES            2
#  define CONF_CLOCK_CPU_DIVIDER                  SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBA_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBB_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1

/* SYSTEM_CLOCK_SOURCE_OSC8M configuration - Internal 8MHz oscillator */
#  define CONF_CLOCK_OSC8M_PRESCALER              SYSTEM_OSC8M_DIV_1
#  define CONF_CLOCK_OSC8M_ON_DEMAND              true
#  define CONF_CLOCK_OSC8M_RUN_IN_STANDBY         false

/* SYSTEM_CLOCK_SOURCE_XOSC configuration - External clock/oscillator */
#  define CONF_CLOCK_XOSC_ENABLE                  false
#  define CONF_CLOCK_XOSC_EXTERNAL_CRYSTAL        SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC_EXTERNAL_FREQUENCY      12000000UL
#  define CONF_CLOCK_XOSC_STARTUP_TIME            SYSTEM_XOSC_STARTUP_32768
#  define CONF_CLOCK_XOSC_AUTO_GAIN_CONTROL       true
#  define CONF_CLOCK_XOSC_ON_DEMAND               true
#  define CONF_CLOCK_XOSC_RUN_IN_STANDBY          false

/* SYSTEM_CLOCK_SOURCE_XOSC32K configuration - External 32KHz crystal/clock oscillator */
#  define CONF_CLOCK_XOSC32K_ENABLE               false
#  define CONF_CLOCK_XOSC32K_EXTERNAL_CRYSTAL     SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC32K_STARTUP_TIME         SYSTEM_XOSC32K_STARTUP_65536
#  define CONF_CLOCK_XOSC32K_AUTO_AMPLITUDE_CONTROL  false
#  define CONF_CLOCK_XOSC32K_ENABLE_1KHZ_OUPUT    false
#  define CONF_CLOCK_XOSC32K_ENABLE_32KHZ_OUTPUT  true
#  define CONF_CLOCK_XOSC32K_ON_DEMAND            true
#  define CONF_CLOCK_XOSC32K_RUN_IN_STANDBY       false

/* SYSTEM_CLOCK_SOURCE_OSC32K configuration - Internal 32KHz oscillator */
#  define CONF_CLOCK_OSC32K_ENABLE                false
#  define CONF_CLOCK_OSC32K_STARTUP_TIME          SYSTEM_OSC32K_STARTUP_130
#  define CONF_CLOCK_OSC32K_ENABLE_1KHZ_OUTPUT    true
```

```
#  define CONF_CLOCK_OSC32K_ENABLE_32KHZ_OUTPUT   true
#  define CONF_CLOCK_OSC32K_ON_DEMAND             true
#  define CONF_CLOCK_OSC32K_RUN_IN_STANDBY        false

/* SYSTEM_CLOCK_SOURCE_DFLL configuration - Digital Frequency Locked Loop */
#  define CONF_CLOCK_DFLL_ENABLE                  true
#  define CONF_CLOCK_DFLL_LOOP_MODE               SYSTEM_CLOCK_DFLL_LOOP_MODE_USB_RECOVERY
#  define CONF_CLOCK_DFLL_ON_DEMAND               true

/* DFLL open loop mode configuration */
#  define CONF_CLOCK_DFLL_COARSE_VALUE            (0x1f / 4)
#  define CONF_CLOCK_DFLL_FINE_VALUE              (0xff / 4)

/* DFLL closed loop mode configuration */
#  define CONF_CLOCK_DFLL_SOURCE_GCLK_GENERATOR   GCLK_GENERATOR_1
#  define CONF_CLOCK_DFLL_MULTIPLY_FACTOR         (48000000 / 32768)
#  define CONF_CLOCK_DFLL_QUICK_LOCK              true
#  define CONF_CLOCK_DFLL_TRACK_AFTER_FINE_LOCK   true
#  define CONF_CLOCK_DFLL_KEEP_LOCK_ON_WAKEUP     true
#  define CONF_CLOCK_DFLL_ENABLE_CHILL_CYCLE      true
#  define CONF_CLOCK_DFLL_MAX_COARSE_STEP_SIZE    (0x1f / 4)
#  define CONF_CLOCK_DFLL_MAX_FINE_STEP_SIZE      (0xff / 4)

/* SYSTEM_CLOCK_SOURCE_DPLL configuration - Digital Phase-Locked Loop */
#  define CONF_CLOCK_DPLL_ENABLE                  false
#  define CONF_CLOCK_DPLL_ON_DEMAND               true
#  define CONF_CLOCK_DPLL_RUN_IN_STANDBY          false
#  define CONF_CLOCK_DPLL_LOCK_BYPASS             false
#  define CONF_CLOCK_DPLL_WAKE_UP_FAST            false
#  define CONF_CLOCK_DPLL_LOW_POWER_ENABLE        false

#  define CONF_CLOCK_DPLL_LOCK_TIME               SYSTEM_CLOCK_SOURCE_DPLL_LOCK_TIME_NO_TIMEOUT
#  define CONF_CLOCK_DPLL_REFERENCE_CLOCK         SYSTEM_CLOCK_SOURCE_DPLL_REFERENCE_CLOCK_REF0
#  define CONF_CLOCK_DPLL_FILTER                  SYSTEM_CLOCK_SOURCE_DPLL_FILTER_DEFAULT

#  define CONF_CLOCK_DPLL_REFERENCE_FREQUENCY     32768
#  define CONF_CLOCK_DPLL_REFEREMCE_DIVIDER       1
#  define CONF_CLOCK_DPLL_OUTPUT_FREQUENCY        48000000

/* Set this to true to configure the GCLK when running clocks_init. If set to
 * false, none of the GCLK generators will be configured in clocks_init(). */
#  define CONF_CLOCK_CONFIGURE_GCLK               true

/* Configure GCLK generator 0 (Main Clock) */
#  define CONF_CLOCK_GCLK_0_ENABLE                true
#  define CONF_CLOCK_GCLK_0_RUN_IN_STANDBY        true
#  define CONF_CLOCK_GCLK_0_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_DFLL
#  define CONF_CLOCK_GCLK_0_PRESCALER             1
#  define CONF_CLOCK_GCLK_0_OUTPUT_ENABLE         false

/* Configure GCLK generator 1 */
#  define CONF_CLOCK_GCLK_1_ENABLE                false
#  define CONF_CLOCK_GCLK_1_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_1_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_XOSC32K
#  define CONF_CLOCK_GCLK_1_PRESCALER             1
#  define CONF_CLOCK_GCLK_1_OUTPUT_ENABLE         false

/* Configure GCLK generator 2 (RTC) */
#  define CONF_CLOCK_GCLK_2_ENABLE                false
#  define CONF_CLOCK_GCLK_2_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_2_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC32K
```

```
#  define CONF_CLOCK_GCLK_2_PRESCALER             32
#  define CONF_CLOCK_GCLK_2_OUTPUT_ENABLE         false

/* Configure GCLK generator 3 */
#  define CONF_CLOCK_GCLK_3_ENABLE                false
#  define CONF_CLOCK_GCLK_3_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_3_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_3_PRESCALER             1
#  define CONF_CLOCK_GCLK_3_OUTPUT_ENABLE         false

/* Configure GCLK generator 4 */
#  define CONF_CLOCK_GCLK_4_ENABLE                false
#  define CONF_CLOCK_GCLK_4_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_4_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_4_PRESCALER             1
#  define CONF_CLOCK_GCLK_4_OUTPUT_ENABLE         false

/* Configure GCLK generator 5 */
#  define CONF_CLOCK_GCLK_5_ENABLE                false
#  define CONF_CLOCK_GCLK_5_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_5_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_5_PRESCALER             1
#  define CONF_CLOCK_GCLK_5_OUTPUT_ENABLE         false

/* Configure GCLK generator 6 */
#  define CONF_CLOCK_GCLK_6_ENABLE                false
#  define CONF_CLOCK_GCLK_6_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_6_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_6_PRESCALER             1
#  define CONF_CLOCK_GCLK_6_OUTPUT_ENABLE         false

/* Configure GCLK generator 7 */
#  define CONF_CLOCK_GCLK_7_ENABLE                false
#  define CONF_CLOCK_GCLK_7_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_7_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_7_PRESCALER             1
#  define CONF_CLOCK_GCLK_7_OUTPUT_ENABLE         false

#endif /* CONF_CLOCKS_H_INCLUDED */
```

5.3.4 conf_board.h

5.3.4.1 AT32UC3A0, AT32UC3A1, AT32UC3B Devices (USBB)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

5.3.4.2 AT32UC3A3, AT32UC3A4 Devices (USBB with High Speed Support)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example
```

```
#endif /* CONF_BOARD_H_INCLUDED */
```

5.3.4.3 AT32UC3C, ATUCXXD, ATUCXXL3U, ATUCXXL4U Devices (USBC)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

5.3.4.4 SAM3X, SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// USB pins are used
#define CONF_BOARD_USB_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

5.3.4.5 SAMD21 Device (USB)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

/* Enable USB VBUS detect */
#define CONF_BOARD_USB_VBUS_DETECT

#endif /* CONF_BOARD_H_INCLUDED */
```

6. USB Device Interface (UDI) for Mass Storage Class (MSC)

USB Device Interface (UDI) for Mass Storage Class (MSC) provides an interface for the configuration and management of USB MSC storage device.

The outline of this documentation is as follows:

- API Overview
- Quick Start Guide for USB Device Mass Storage Module (UDI MSC)
- Configuration File Examples

For more details for Atmel® Software Framework (ASF) USB Device Stack, refer to following application notes:

- AVR4900: ASF - USB Device Stack[1]
- AVR4920: ASF - USB Device Stack - Compliance and Performance Figures[2]
- AVR4921: ASF - USB Device Stack Differences between ASF V1 and V2[3]

6.1 API Overview

6.1.1 Variable and Type Definitions

6.1.1.1 UDI MSC Interface for UDC

Variable udi_api_msc

```
UDC_DESC_STORAGE udi_api_t udi_api_msc
```

Global structure which contains standard UDI interface for UDC.

6.1.2 Structure Definitions

6.1.2.1 Struct udi_msc_desc_t

Interface descriptor structure for MSC.

Table 6-1. Members

Type	Name	Description
usb_ep_desc_t	ep_in	Data IN endpoint descriptors
usb_ep_desc_t	ep_out	Data OUT endpoint descriptors
usb_iface_desc_t	iface	Standard USB interface descriptor structure

6.1.3 Macro Definitions

6.1.3.1 USB Interface Descriptors

The following structures provide predefined USB interface descriptors. It must be used to define the final USB descriptors.

Macro UDI_MSC_STRING_ID

[1] http://www.atmel.com/dyn/resources/prod_documents/doc8360.pdf
[2] http://www.atmel.com/dyn/resources/prod_documents/doc8410.pdf
[3] http://www.atmel.com/dyn/resources/prod_documents/doc8411.pdf

```
#define UDI_MSC_STRING_ID 0
```

By default no string is associated to this interface.

Macro UDI_MSC_EPS_SIZE_FS

```
#define UDI_MSC_EPS_SIZE_FS 64
```

MSC endpoints size for full speed.

Macro UDI_MSC_EPS_SIZE_HS

```
#define UDI_MSC_EPS_SIZE_HS 512
```

MSC endpoints size for high speed.

Macro UDI_MSC_DESC

```
#define UDI_MSC_DESC \
  .iface.bLength              = sizeof(usb_iface_desc_t),\
   .iface.bDescriptorType     = USB_DT_INTERFACE,\
   .iface.bInterfaceNumber    = UDI_MSC_IFACE_NUMBER,\
   .iface.bAlternateSetting   = 0,\
   .iface.bNumEndpoints       = 2,\
   .iface.bInterfaceClass     = MSC_CLASS,\
   .iface.bInterfaceSubClass  = MSC_SUBCLASS_TRANSPARENT,\
   .iface.bInterfaceProtocol  = MSC_PROTOCOL_BULK,\
   .iface.iInterface          = UDI_MSC_STRING_ID,\
   .ep_in.bLength             = sizeof(usb_ep_desc_t),\
   .ep_in.bDescriptorType     = USB_DT_ENDPOINT,\
   .ep_in.bEndpointAddress    = UDI_MSC_EP_IN,\
   .ep_in.bmAttributes        = USB_EP_TYPE_BULK,\
   .ep_in.bInterval           = 0,\
   .ep_out.bLength            = sizeof(usb_ep_desc_t),\
   .ep_out.bDescriptorType    = USB_DT_ENDPOINT,\
   .ep_out.bEndpointAddress   = UDI_MSC_EP_OUT,\
   .ep_out.bmAttributes       = USB_EP_TYPE_BULK,\
   .ep_out.bInterval          = 0,
```

Content of MSC interface descriptor for all speeds.

Macro UDI_MSC_DESC_FS

```
#define UDI_MSC_DESC_FS \
  {\
   UDI_MSC_DESC \
   .ep_in.wMaxPacketSize      = LE16(UDI_MSC_EPS_SIZE_FS),\
   .ep_out.wMaxPacketSize     = LE16(UDI_MSC_EPS_SIZE_FS),\
   }
```

Content of MSC interface descriptor for full speed only.

Macro UDI_MSC_DESC_HS

```
#define UDI_MSC_DESC_HS \
  {\
   UDI_MSC_DESC \
   .ep_in.wMaxPacketSize      = LE16(UDI_MSC_EPS_SIZE_HS),\
   .ep_out.wMaxPacketSize     = LE16(UDI_MSC_EPS_SIZE_HS),\
   }
```

Content of MSC interface descriptor for high speed only.

6.1.4 Function Definitions

6.1.4.1 Function udi_msc_process_trans()

Process the background read/write commands.

```
bool udi_msc_process_trans(void)
```

Routine called by the main loop.

6.1.4.2 Function udi_msc_trans_block()

Transfers data to/from USB MSC endpoints.

```
bool udi_msc_trans_block(
  bool b_read,
  uint8_t * block,
  iram_size_t block_size,
  void(*)(udd_ep_status_t status, iram_size_t n, udd_ep_id_t ep) callback)
```

Table 6-2. Parameters

Data direction	Parameter name	Description
[in]	b_read	Memory to USB, if true
[in, out]	block	Buffer on Internal RAM to send or fill
[in]	block_size	Buffer size to send or fill
[in]	callback	Function to call at the end of transfer. If NULL then the routine exit when transfer is finish.

Returns 1 if function was successfully done, otherwise 0.

6.2 Quick Start Guide for USB Device Mass Storage Module (UDI MSC)

This is the quick start guide for the USB Device Interface MSC Module (UDI MSC) with step-by-step instructions on how to configure and use the modules in a selection of use cases.

The use cases contain several code fragments. The code fragments in the steps for setup can be copied into a custom initialization function, while the steps for usage can be copied into, e.g., the main application function.

6.2.1 Basic Use Case

In this basic use case, the "USB MSC (Single Interface Device)" module is used. The "USB MSC (Composite Device)" module usage is described in Advanced Use Cases.

6.2.2 Setup Steps

As a USB device, it follows common USB device setup steps. Refer to USB Device Basic Setup.

The USB MSC interface accesses Memory through Common Abstraction Layer (ctrl_access) in ASF. See Common Abstraction Layer for Memory Interfaces.

6.2.2.1 Common Abstraction Layer for Memory Interfaces

Common abstraction layer (ctrl_access) can provide interfaces between Memory and USB. In USB MSC UDI the read/write invokes following ctrl_access functions:

```
extern Ctrl_status memory_2_usb(U8 lun, U32 addr, U16 nb_sector);
extern Ctrl_status usb_2_memory(U8 lun, U32 addr, U16 nb_sector);
```

Then the ctrl_access dispatch the read/write operation to different LUNs.

The memory access in ctrl_access is configured through conf_access.h. E.g., to use LUN0 to access virtual memory disk, the configuration should include:

```
#define LUN_0                ENABLE // Enable LUN0 access
//...

#define VIRTUAL_MEM          LUN_0
#define LUN_ID_VIRTUAL_MEM   LUN_ID_0
#define LUN_0_INCLUDE        "virtual_mem.h"         // APIs (complied to ctrl_access)
#define Lun_0_test_unit_ready virtual_test_unit_ready // check disk ready
#define Lun_0_read_capacity  virtual_read_capacity   // get disk size
#define Lun_0_wr_protect     virtual_wr_protect      // check protection
#define Lun_0_removal        virtual_removal         // check if disk is removable
#define Lun_0_usb_read_10    virtual_usb_read_10     // Disk to USB transfer
#define Lun_0_usb_write_10   virtual_usb_write_10    // USB to Disk transfer
#define LUN_0_NAME           "\"On-Chip Virtual Memory\""
//...

#define ACCESS_USB               true    // USB interface.
//...

#define GLOBAL_WR_PROTECT        false
```

Since LUN_0 is defined as a "Virtual Memory", the module to encapsulate the internal or on-board memory to access as a disk is included. The configuration of such a virtual memory disk is in conf_virtual_mem.h. E.g., to use internal RAM to build such a memory disk, the configuration should include:

```
#define VMEM_NB_SECTOR 48 //Internal RAM 24KB (should > 20KB or PC can not format it)
```

For more examples of the control access or disk configuration, refer to conf_access.h and conf_virtual_mem.h.

For more Information about Memory Control Access, refer to the online document:

- Atmel Software Framework - Memory Control Access[4]

[4] http://asf.atmel.com/docs/latest/sam3a/html/group__group__common__services__storage__ctrl__access.html

6.2.3 Usage Steps

6.2.3.1 Example Code

Content of conf_usb.h:

```
#define  USB_DEVICE_SERIAL_NAME  "12...EF" // Disk SN for MSC
#define UDI_MSC_GLOBAL_VENDOR_ID \
   'A', 'T', 'M', 'E', 'L', ' ', ' ', ' '
#define UDI_MSC_GLOBAL_PRODUCT_VERSION \
   '1', '.', '0', '0'
#define UDI_MSC_ENABLE_EXT() my_callback_msc_enable()
extern bool my_callback_msc_enable(void);
#define UDI_MSC_DISABLE_EXT() my_callback_msc_disable()
extern void my_callback_msc_disable(void);
#include "udi_msc_conf.h" // At the end of conf_usb.h file
```

Add to application C-file:

```
static bool my_flag_autorize_msc_transfert = false;
bool my_callback_msc_enable(void)
{
   my_flag_autorize_msc_transfert = true;
   return true;
}
void my_callback_msc_disable(void)
{
   my_flag_autorize_msc_transfert = false;
}

void task(void)
{
   udi_msc_process_trans();
}
```

6.2.3.2 Workflow

1. Ensure that conf_usb.h is available and contains the following configuration which is the USB device MSC configuration:

```
#define USB_DEVICE_SERIAL_NAME  "12...EF" // Disk SN for MSC
```

Note The USB serial number is mandatory when a MSC interface is used.

```
#define UDI_MSC_GLOBAL_VENDOR_ID \
   'A', 'T', 'M', 'E', 'L', ' ', ' ', ' '
#define UDI_MSC_GLOBAL_PRODUCT_VERSION \
   '1', '.', '0', '0'
```

Note The USB MSC interface requires a vendor ID (eight ASCII characters) and a product version (four ASCII characters).

```
#define UDI_MSC_ENABLE_EXT() my_callback_msc_enable()
```

```
extern bool my_callback_msc_enable(void);
```

Note After the device enumeration (detecting and identifying USB devices), the USB host starts the device configuration. When the USB MSC interface from the device is accepted by the host, the USB host enables this interface and the UDI_MSC_ENABLE_EXT() callback function is called and return true. Thus, when this event is received, the tasks which call udi_msc_process_trans() must be enabled.

```
#define UDI_MSC_DISABLE_EXT() my_callback_msc_disable()
extern void my_callback_msc_disable(void);
```

Note When the USB device is unplugged or is reset by the USB host, the USB interface is disabled and the UDI_MSC_DISABLE_EXT() callback function is called. Thus, it is recommended to disable the task which is called udi_msc_process_trans().

2. The MSC is automatically linked with memory control access component which provides the memories interfaces. However, the memory data transfers must be done outside USB interrupt routine. This is done in the MSC process ("udi_msc_process_trans()") called by main loop:

```
void task(void) {
  udi_msc_process_trans();
}
```

3. The MSC speed depends on task periodicity. To get the best speed the notification callback "UDI_MSC_NOTIFY_TRANS_EXT" can be used to wakeup this task (Example, through a mutex):

```
#define  UDI_MSC_NOTIFY_TRANS_EXT()    msc_notify_trans()
void msc_notify_trans(void) {
  wakeup_my_task();
}
```

6.2.4 Advanced Use Cases

For multiple interface use of UDI MSC module, see the following:

- MSC in a Composite Device

For more advanced use of the UDI CDC module, see the following:

- USB Device Advanced Use Cases.

6.2.5 MSC in a Composite Device

A USB Composite Device is a USB Device which uses more than one USB class. In this use case, the "USB MSC (Composite Device)" module is used to create a USB composite device. Thus, this USB module can be associated with another "Composite Device" module, like "USB HID Mouse (Composite Device)".

Also, you can refer to application note AVR4902 ASF - USB Composite Device[5].

6.2.5.1 Setup Steps

For the setup code of this use case to work, the Basic Use Case must be followed.

[5] http://www.atmel.com/dyn/resources/prod_documents/doc8445.pdf

6.2.5.2 Usage Steps

Example Code

Content of conf_usb.h:

```
#define USB_DEVICE_EP_CTRL_SIZE  64
#define USB_DEVICE_NB_INTERFACE (X+1)
#define USB_DEVICE_MAX_EP (X+2)

#define UDI_MSC_EP_IN  (X | USB_EP_DIR_IN)
#define UDI_MSC_EP_OUT (Y | USB_EP_DIR_OUT)
#define UDI_MSC_IFACE_NUMBER  X

#define UDI_COMPOSITE_DESC_T \
   udi_msc_desc_t udi_msc; \
   ...
#define UDI_COMPOSITE_DESC_FS \
   .udi_msc = UDI_MSC_DESC, \
   ...
#define UDI_COMPOSITE_DESC_HS \
   .udi_msc = UDI_MSC_DESC, \
   ...
#define UDI_COMPOSITE_API \
   &udi_api_msc, \
   ...
```

Workflow

1. Ensure that conf_usb.h is available and contains the following parameters required for a USB composite device configuration:

```
// Endpoint control size, This must be:
// - 8, 16, 32 or 64 for full speed device (8 is recommended to save RAM)
// - 64 for a high speed device
#define USB_DEVICE_EP_CTRL_SIZE  64
// Total Number of interfaces on this USB device.
// Add 1 for MSC.
#define USB_DEVICE_NB_INTERFACE (X+1)
// Total number of endpoints on this USB device.
// This must include each endpoint for each interface.
// Add 2 for MSC.
#define USB_DEVICE_MAX_EP (X+2)
```

2. Ensure that conf_usb.h contains the description of composite device:

```
// The endpoint numbers chosen by you for the MSC.
// The endpoint numbers starting from 1.
#define UDI_MSC_EP_IN  (X | USB_EP_DIR_IN)
#define UDI_MSC_EP_OUT (Y | USB_EP_DIR_OUT)
// The interface index of an interface starting from 0
#define UDI_MSC_IFACE_NUMBER  X
```

3. Ensure that conf_usb.h contains the following parameters required for a USB composite device configuration:

```
// USB Interfaces descriptor structure
#define UDI_COMPOSITE_DESC_T \
   ...
```

```
   udi_msc_desc_t udi_msc; \
   ...
// USB Interfaces descriptor value for Full Speed
#define UDI_COMPOSITE_DESC_FS \
   ...
   .udi_msc = UDI_MSC_DESC_FS, \
   ...
// USB Interfaces descriptor value for High Speed
#define UDI_COMPOSITE_DESC_HS \
   ...
   .udi_msc = UDI_MSC_DESC_HS, \
   ...
// USB Interface APIs
#define UDI_COMPOSITE_API \
   ...
   &udi_api_msc, \
   ...
```

Note The descriptors order given in the four lists above must be the same as the order defined by all interface indexes. The interface index orders are defined through UDI_X_IFACE_NUMBER defines.

6.3 Configuration File Examples

6.3.1 conf_usb.h

6.3.1.1 UDI MSC Single

```
#ifndef _CONF_USB_H_
#define _CONF_USB_H_

#include "compiler.h"

#warning You must refill the following definitions with a correct values

#define  USB_DEVICE_VENDOR_ID             USB_VID_ATMEL
#define  USB_DEVICE_PRODUCT_ID            USB_PID_ATMEL_ASF_HIDGENERIC
#define  USB_DEVICE_MAJOR_VERSION         1
#define  USB_DEVICE_MINOR_VERSION         0
#define  USB_DEVICE_POWER                 100 // Consumption on Vbus line (mA)
#define  USB_DEVICE_ATTR                  \
    (USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_BUS_POWERED)
//  (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_SELF_POWERED)
//  (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_BUS_POWERED)

// #define  USB_DEVICE_MANUFACTURE_NAME      "Manufacture name"
// #define  USB_DEVICE_PRODUCT_NAME          "Product name"
// #define  USB_DEVICE_SERIAL_NAME           "12...EF"

//#define  USB_DEVICE_LOW_SPEED

#if (UC3A3||UC3A4)
//#define  USB_DEVICE_HS_SUPPORT
#endif


// #define  UDC_VBUS_EVENT(b_vbus_high)      user_callback_vbus_action(b_vbus_high)
```

```
// extern void user_callback_vbus_action(bool b_vbus_high);
// #define  UDC_SOF_EVENT()                  user_callback_sof_action()
// extern void user_callback_sof_action(void);
// #define  UDC_SUSPEND_EVENT()              user_callback_suspend_action()
// extern void user_callback_suspend_action(void);
// #define  UDC_RESUME_EVENT()               user_callback_resume_action()
// extern void user_callback_resume_action(void);
// #define  UDC_REMOTEWAKEUP_ENABLE()        user_callback_remotewakeup_enable()
// extern void user_callback_remotewakeup_enable(void);
// #define  UDC_REMOTEWAKEUP_DISABLE()       user_callback_remotewakeup_disable()
// extern void user_callback_remotewakeup_disable(void);
// #define  UDC_GET_EXTRA_STRING()



#define  UDI_HID_GENERIC_ENABLE_EXT()       true
#define  UDI_HID_GENERIC_DISABLE_EXT()
#define  UDI_HID_GENERIC_REPORT_OUT(ptr)
#define  UDI_HID_GENERIC_SET_FEATURE(f)
/*
 * #define UDI_HID_GENERIC_ENABLE_EXT() my_callback_generic_enable()
 * extern bool my_callback_generic_enable(void);
 * #define UDI_HID_GENERIC_DISABLE_EXT() my_callback_generic_disable()
 * extern void my_callback_generic_disable(void);
 * #define  UDI_HID_GENERIC_REPORT_OUT(ptr) my_callback_generic_report_out(ptr)
 * extern void my_callback_generic_report_out(uint8_t *report);
 * #define  UDI_HID_GENERIC_SET_FEATURE(f) my_callback_generic_set_feature(f)
 * extern void my_callback_generic_set_feature(uint8_t *report_feature);
 */

#define  UDI_HID_REPORT_IN_SIZE             64
#define  UDI_HID_REPORT_OUT_SIZE            64
#define  UDI_HID_REPORT_FEATURE_SIZE        4

#define  UDI_HID_GENERIC_EP_SIZE            64




#include "udi_hid_generic_conf.h"

#endif // _CONF_USB_H_
```

6.3.1.2 UDI MSC Multiple (Composite)

```
#ifndef _CONF_USB_H_
#define _CONF_USB_H_

#include "compiler.h"

#warning You must refill the following definitions with a correct values

#define  USB_DEVICE_VENDOR_ID             USB_VID_ATMEL
#define  USB_DEVICE_PRODUCT_ID            0xFFFF
#define  USB_DEVICE_MAJOR_VERSION         1
#define  USB_DEVICE_MINOR_VERSION         0
#define  USB_DEVICE_POWER                 100 // Consumption on VBUS line (mA)
#define  USB_DEVICE_ATTR                  \
        (USB_CONFIG_ATTR_SELF_POWERED)
```

```
// (USB_CONFIG_ATTR_BUS_POWERED)
// (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_BUS_POWERED)

// #define  USB_DEVICE_MANUFACTURE_NAME      "Manufacture name"
// #define  USB_DEVICE_PRODUCT_NAME          "Product name"
// #define  USB_DEVICE_SERIAL_NAME           "12...EF" // Disk SN for MSC

//#define  USB_DEVICE_LOW_SPEED

#if (UC3A3||UC3A4)
//#define  USB_DEVICE_HS_SUPPORT
#endif


// #define  UDC_VBUS_EVENT(b_vbus_high)      user_callback_vbus_action(b_vbus_high)
// extern void user_callback_vbus_action(bool b_vbus_high);
// #define  UDC_SOF_EVENT()                  user_callback_sof_action()
// extern void user_callback_sof_action(void);
// #define  UDC_SUSPEND_EVENT()              user_callback_suspend_action()
// extern void user_callback_suspend_action(void);
// #define  UDC_RESUME_EVENT()               user_callback_resume_action()
// extern void user_callback_resume_action(void);
// #define  UDC_REMOTEWAKEUP_ENABLE()        user_callback_remotewakeup_enable()
// extern void user_callback_remotewakeup_enable(void);
// #define  UDC_REMOTEWAKEUP_DISABLE()       user_callback_remotewakeup_disable()
// extern void user_callback_remotewakeup_disable(void);
// #define  UDC_GET_EXTRA_STRING()

#define  USB_DEVICE_EP_CTRL_SIZE       64

#define  USB_DEVICE_NB_INTERFACE       1 // 1 or more

#define  USB_DEVICE_MAX_EP             1 // 0 to max endpoint requested by interfaces



#define  UDI_CDC_PORT_NB 1

#define  UDI_CDC_ENABLE_EXT(port)             true
#define  UDI_CDC_DISABLE_EXT(port)
#define  UDI_CDC_RX_NOTIFY(port)
#define  UDI_CDC_TX_EMPTY_NOTIFY(port)
#define  UDI_CDC_SET_CODING_EXT(port,cfg)
#define  UDI_CDC_SET_DTR_EXT(port,set)
#define  UDI_CDC_SET_RTS_EXT(port,set)
/*
 * #define UDI_CDC_ENABLE_EXT(port) my_callback_cdc_enable()
 * extern bool my_callback_cdc_enable(void);
 * #define UDI_CDC_DISABLE_EXT(port) my_callback_cdc_disable()
 * extern void my_callback_cdc_disable(void);
 * #define  UDI_CDC_RX_NOTIFY(port) my_callback_rx_notify(port)
 * extern void my_callback_rx_notify(uint8_t port);
 * #define  UDI_CDC_TX_EMPTY_NOTIFY(port) my_callback_tx_empty_notify(port)
 * extern void my_callback_tx_empty_notify(uint8_t port);
 * #define  UDI_CDC_SET_CODING_EXT(port,cfg) my_callback_config(port,cfg)
 * extern void my_callback_config(uint8_t port, usb_cdc_line_coding_t * cfg);
 * #define  UDI_CDC_SET_DTR_EXT(port,set) my_callback_cdc_set_dtr(port,set)
 * extern void my_callback_cdc_set_dtr(uint8_t port, bool b_enable);
 * #define  UDI_CDC_SET_RTS_EXT(port,set) my_callback_cdc_set_rts(port,set)
```

```
 * extern void my_callback_cdc_set_rts(uint8_t port, bool b_enable);
 */

#define  UDI_CDC_LOW_RATE

#define  UDI_CDC_DEFAULT_RATE             115200
#define  UDI_CDC_DEFAULT_STOPBITS         CDC_STOP_BITS_1
#define  UDI_CDC_DEFAULT_PARITY           CDC_PAR_NONE
#define  UDI_CDC_DEFAULT_DATABITS         8

#define  UDI_CDC_DATA_EP_IN_0          (1 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_0         (2 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_0             (3 | USB_EP_DIR_IN)  // Notify endpoint
#define  UDI_CDC_DATA_EP_IN_2          (4 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_2         (5 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_2             (6 | USB_EP_DIR_IN)  // Notify endpoint
#define  UDI_CDC_DATA_EP_IN_3          (7 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_3         (8 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_3             (9 | USB_EP_DIR_IN)  // Notify endpoint

#define  UDI_CDC_COMM_IFACE_NUMBER_0   0
#define  UDI_CDC_DATA_IFACE_NUMBER_0   1
#define  UDI_CDC_COMM_IFACE_NUMBER_2   2
#define  UDI_CDC_DATA_IFACE_NUMBER_2   3
#define  UDI_CDC_COMM_IFACE_NUMBER_3   4
#define  UDI_CDC_DATA_IFACE_NUMBER_3   5




#define UDI_MSC_GLOBAL_VENDOR_ID            \
   'A', 'T', 'M', 'E', 'L', ' ', ' ', ' '
#define UDI_MSC_GLOBAL_PRODUCT_VERSION           \
   '1', '.', '0', '0'

#define  UDI_MSC_ENABLE_EXT()          true
#define  UDI_MSC_DISABLE_EXT()
#define  UDI_MSC_NOTIFY_TRANS_EXT()
/*
 * #define UDI_MSC_ENABLE_EXT() my_callback_msc_enable()
 * extern bool my_callback_msc_enable(void);
 * #define UDI_MSC_DISABLE_EXT() my_callback_msc_disable()
 * extern void my_callback_msc_disable(void);
 * #define  UDI_MSC_NOTIFY_TRANS_EXT()    msc_notify_trans()
 * extern void msc_notify_trans(void) {
 */

#define  UDI_MSC_EP_IN                 (1 | USB_EP_DIR_IN)
#define  UDI_MSC_EP_OUT                (2 | USB_EP_DIR_OUT)

#define  UDI_MSC_IFACE_NUMBER          0




#define  UDI_HID_MOUSE_ENABLE_EXT()       true
#define  UDI_HID_MOUSE_DISABLE_EXT()
// #define UDI_HID_MOUSE_ENABLE_EXT() my_callback_mouse_enable()
// extern bool my_callback_mouse_enable(void);
// #define UDI_HID_MOUSE_DISABLE_EXT() my_callback_mouse_disable()
// extern void my_callback_mouse_disable(void);
```

```
#define  UDI_HID_MOUSE_EP_IN           (1 | USB_EP_DIR_IN)

#define  UDI_HID_MOUSE_IFACE_NUMBER    0



#define  UDI_HID_KBD_ENABLE_EXT()       true
#define  UDI_HID_KBD_DISABLE_EXT()
// #define UDI_HID_KBD_ENABLE_EXT() my_callback_keyboard_enable()
// extern bool my_callback_keyboard_enable(void);
// #define UDI_HID_KBD_DISABLE_EXT() my_callback_keyboard_disable()
// extern void my_callback_keyboard_disable(void);
#define  UDI_HID_KBD_CHANGE_LED(value)
// #define  UDI_HID_KBD_CHANGE_LED(value) my_callback_keyboard_led(value)
// extern void my_callback_keyboard_led(uint8_t value)

#define  UDI_HID_KBD_EP_IN           (1 | USB_EP_DIR_IN)

#define  UDI_HID_KBD_IFACE_NUMBER    0



#define  UDI_HID_GENERIC_ENABLE_EXT()       true
#define  UDI_HID_GENERIC_DISABLE_EXT()
#define  UDI_HID_GENERIC_REPORT_OUT(ptr)
#define  UDI_HID_GENERIC_SET_FEATURE(f)
/*
 * #define UDI_HID_GENERIC_ENABLE_EXT() my_callback_generic_enable()
 * extern bool my_callback_generic_enable(void);
 * #define UDI_HID_GENERIC_DISABLE_EXT() my_callback_generic_disable()
 * extern void my_callback_generic_disable(void);
 * #define  UDI_HID_GENERIC_REPORT_OUT(ptr) my_callback_generic_report_out(ptr)
 * extern void my_callback_generic_report_out(uint8_t *report);
 * #define  UDI_HID_GENERIC_SET_FEATURE(f) my_callback_generic_set_feature(f)
 * extern void my_callback_generic_set_feature(uint8_t *report_feature);
 */
#define  UDI_HID_REPORT_IN_SIZE             64
#define  UDI_HID_REPORT_OUT_SIZE            64
#define  UDI_HID_REPORT_FEATURE_SIZE        4
#define  UDI_HID_GENERIC_EP_SIZE            64

#define  UDI_HID_GENERIC_EP_OUT   (2 | USB_EP_DIR_OUT)
#define  UDI_HID_GENERIC_EP_IN    (1 | USB_EP_DIR_IN)

#define  UDI_HID_GENERIC_IFACE_NUMBER    0



#define  UDI_PHDC_ENABLE_EXT()          true
#define  UDI_PHDC_DISABLE_EXT()

#define  UDI_PHDC_DATAMSG_FORMAT       USB_PHDC_DATAMSG_FORMAT_11073_20601
#define  UDI_PHDC_SPECIALIZATION       {0x2345} // Define in 11073_20601

#define  UDI_PHDC_QOS_OUT              \
        (USB_PHDC_QOS_MEDIUM_BETTER|USB_PHDC_QOS_HIGH_BEST)
#define  UDI_PHDC_QOS_IN               \
```

```
        (USB_PHDC_QOS_LOW_GOOD|USB_PHDC_QOS_MEDIUM_BETTER|USB_PHDC_QOS_MEDIUM_BEST)

#define  UDI_PHDC_METADATA_DESC_BULK_IN   {0x01,0x02,0x03}
#define  UDI_PHDC_METADATA_DESC_BULK_OUT  {0x01,0x02,0x03}
#define  UDI_PHDC_METADATA_DESC_INT_IN    {0x01,0x02,0x03}

#define  UDI_PHDC_EP_BULK_OUT          (1 | USB_EP_DIR_OUT)
#define  UDI_PHDC_EP_BULK_IN           (2 | USB_EP_DIR_IN)
#if ((UDI_PHDC_QOS_IN&USB_PHDC_QOS_LOW_GOOD)==USB_PHDC_QOS_LOW_GOOD)
// Only if UDI_PHDC_QOS_IN include USB_PHDC_QOS_LOW_GOOD
#  define  UDI_PHDC_EP_INTERRUPT_IN      (3 | USB_EP_DIR_IN)
#endif

#define  UDI_PHDC_EP_SIZE_BULK_OUT     32
#define  UDI_PHDC_EP_SIZE_BULK_IN      32
#define  UDI_PHDC_EP_SIZE_INT_IN       8


#define  UDI_PHDC_IFACE_NUMBER          0



#define  UDI_VENDOR_ENABLE_EXT()          true
#define  UDI_VENDOR_DISABLE_EXT()
#define  UDI_VENDOR_SETUP_OUT_RECEIVED()  false
#define  UDI_VENDOR_SETUP_IN_RECEIVED()   false
/*
 * #define UDI_VENDOR_ENABLE_EXT() my_callback_vendor_enable()
 * extern bool my_callback_vendor_enable(void);
 * #define UDI_VENDOR_DISABLE_EXT() my_callback_vendor_disable()
 * extern void my_callback_vendor_disable(void);
 *
 * #define  UDI_VENDOR_SETUP_OUT_RECEIVED()  my_vendor_setup_out_received()
 * extern bool my_vendor_setup_out_received(void);
 * #define  UDI_VENDOR_SETUP_IN_RECEIVED()   my_vendor_setup_in_received()
 * extern bool my_vendor_setup_in_received(void);
 */

#define  UDI_VENDOR_EPS_SIZE_INT_FS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_FS   64
#define  UDI_VENDOR_EPS_SIZE_ISO_FS   256

#define  UDI_VENDOR_EPS_SIZE_INT_HS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_HS  512
#define  UDI_VENDOR_EPS_SIZE_ISO_HS    64

#define  UDI_VENDOR_EP_INTERRUPT_IN  (1 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_INTERRUPT_OUT (2 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_BULK_IN       (3 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_BULK_OUT      (4 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_ISO_IN        (5 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_ISO_OUT       (6 | USB_EP_DIR_OUT)


#define  UDI_VENDOR_IFACE_NUMBER     0



//... Eventually add other Interface Configuration
```

```
#define UDI_COMPOSITE_DESC_T

#define UDI_COMPOSITE_DESC_FS

#define UDI_COMPOSITE_DESC_HS

#define UDI_COMPOSITE_API

/* Example for device with cdc, msc and hid mouse interface
#define UDI_COMPOSITE_DESC_T \
    usb_iad_desc_t udi_cdc_iad; \
    udi_cdc_comm_desc_t udi_cdc_comm; \
    udi_cdc_data_desc_t udi_cdc_data; \
    udi_msc_desc_t udi_msc; \
    udi_hid_mouse_desc_t udi_hid_mouse

#define UDI_COMPOSITE_DESC_FS \
    .udi_cdc_iad               = UDI_CDC_IAD_DESC_0, \
    .udi_cdc_comm              = UDI_CDC_COMM_DESC_0, \
    .udi_cdc_data              = UDI_CDC_DATA_DESC_0_FS, \
    .udi_msc                   = UDI_MSC_DESC_FS, \
    .udi_hid_mouse             = UDI_HID_MOUSE_DESC

#define UDI_COMPOSITE_DESC_HS \
    .udi_cdc_iad               = UDI_CDC_IAD_DESC_0, \
    .udi_cdc_comm              = UDI_CDC_COMM_DESC_0, \
    .udi_cdc_data              = UDI_CDC_DATA_DESC_0_HS, \
    .udi_msc                   = UDI_MSC_DESC_HS, \
    .udi_hid_mouse             = UDI_HID_MOUSE_DESC

#define UDI_COMPOSITE_API  \
    &udi_api_cdc_comm,       \
    &udi_api_cdc_data,       \
    &udi_api_msc,            \
    &udi_api_hid_mouse
*/




/* Example of include for interface
#include "udi_msc.h"
#include "udi_hid_kbd.h"
#include "udi_hid_mouse.h"
#include "udi_cdc.h"
#include "udi_phdc.h"
#include "udi_vendor.h"
*/
/* Declaration of callbacks used by USB
#include "callback_def.h"
*/

#endif // _CONF_USB_H_
```

6.3.2 conf_clock.h

6.3.2.1 XMEGA (USB)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

#define CONFIG_USBCLK_SOURCE     USBCLK_SRC_RCOSC
#define CONFIG_OSC_RC32_CAL      48000000UL

#define CONFIG_OSC_AUTOCAL_RC32MHZ_REF_OSC  OSC_ID_USBSOF

#define CONFIG_SYSCLK_SOURCE     SYSCLK_SRC_RC32MHZ
#define CONFIG_SYSCLK_PSADIV     SYSCLK_PSADIV_2
#define CONFIG_SYSCLK_PSBCDIV    SYSCLK_PSBCDIV_1_2

/*

#define CONFIG_PLL0_SOURCE       PLL_SRC_XOSC
#define CONFIG_PLL0_MUL          6
#define CONFIG_PLL0_DIV          1

#define CONFIG_USBCLK_SOURCE     USBCLK_SRC_PLL

#define CONFIG_SYSCLK_SOURCE     SYSCLK_SRC_PLL
#define CONFIG_SYSCLK_PSADIV     SYSCLK_PSADIV_2
#define CONFIG_SYSCLK_PSBCDIV    SYSCLK_PSBCDIV_1_2
*/


#endif /* CONF_CLOCK_H_INCLUDED */
```

6.3.2.2 AT32UC3A0, AT32UC3A1, and AT32UC3B Devices (USBB)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL0

// ===== PLL0 Options
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_MUL             4 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL0_DIV             1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
#define CONFIG_PLL1_SOURCE            PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
#define CONFIG_PLL1_MUL               8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL1_DIV               2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
//#define CONFIG_SYSCLK_CPU_DIV         0 /* Fcpu = Fsys/(2 ^ CPU_div) */
//#define CONFIG_SYSCLK_PBA_DIV         0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV         0 /* Fpbb = Fsys/(2 ^ PBB_div) */

// ===== Peripheral Clock Management Options
```

```
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

6.3.2.3 AT32UC3A3, and AT32UC3A4 Devices (USBB with High Speed Support)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL0

// ===== PLL0 Options
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_MUL             11 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL0_DIV             2  /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL1_MUL             8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL1_DIV             2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
#define CONFIG_SYSCLK_CPU_DIV         0 /* Fcpu = Fsys/(2 ^ CPU_div) */
#define CONFIG_SYSCLK_PBA_DIV         0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV       0 /* Fpbb = Fsys/(2 ^ PBB_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

6.3.2.4 AT32UC3C, ATUCXXD, ATUCXXL3U, and ATUCXXL4U Devices (USBC)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED
```

```
// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC1
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_PLL0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL1
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RC8M

// ===== PLL0 Options
#define CONFIG_PLL0_SOURCE            PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_SOURCE          PLL_SRC_RC8M
#define CONFIG_PLL0_MUL               3 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL0_DIV               1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL1_SOURCE          PLL_SRC_RC8M
//#define CONFIG_PLL1_MUL             3 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL1_DIV             1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
//#define CONFIG_SYSCLK_CPU_DIV       0 /* Fcpu = Fsys/(2 ^ CPU_div) */
//#define CONFIG_SYSCLK_PBA_DIV       0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV       0 /* Fpbb = Fsys/(2 ^ PBB_div) */
//#define CONFIG_SYSCLK_PBC_DIV       0 /* Fpbc = Fsys/(2 ^ PBC_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC1
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

6.3.2.5 SAM3S, SAM3SD, and SAM4S Devices (UPD: USB Peripheral Device)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock (MCK) Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_BYPASS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_4M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_8M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_12M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_BYPASS
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLLACK
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLLBCK
```

```
// ===== System Clock (MCK) Prescaler Options   (Fmck = Fsys / (SYSCLK_PRES))
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_1
#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_2
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_4
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_8
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_16
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_32
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_64
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_3

// ===== PLL0 (A) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
// Use mul and div effective values here.
#define CONFIG_PLL0_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL0_MUL             32
#define CONFIG_PLL0_DIV             3

// ===== PLL1 (B) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
// Use mul and div effective values here.
#define CONFIG_PLL1_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL1_MUL             16
#define CONFIG_PLL1_DIV             2

// ===== USB Clock Source Options   (Fusb = FpllX / USB_div)
// Use div effective value here.
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV           2

// ===== Target frequency (System clock)
// - XTAL frequency: 12MHz
// - System clock source: PLLA
// - System clock prescaler: 2 (divided by 2)
// - PLLA source: XTAL
// - PLLA output: XTAL * 32 / 3
// - System clock is: 12 * 32 / 3 / 2 = 64MHz
// ===== Target frequency (USB Clock)
// - USB clock source: PLLB
// - USB clock divider: 2 (divided by 2)
// - PLLB output: XTAL * 16 / 2
// - USB clock: 12 * 16 / 2 / 2 = 48MHz


#endif /* CONF_CLOCK_H_INCLUDED */
```

6.3.2.6 SAM3U Device (UPDHS: USB Peripheral Device High Speed)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock (MCK) Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_BYPASS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_4M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_8M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_12M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_BYPASS
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLLACK
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_UPLLCK
```

```
// ===== System Clock (MCK) Prescaler Options   (Fmck = Fsys / (SYSCLK_PRES))
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_1
#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_2
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_4
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_8
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_16
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_32
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_64
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_3

// ===== PLL0 (A) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
// Use mul and div effective values here.
#define CONFIG_PLL0_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL0_MUL             16
#define CONFIG_PLL0_DIV             1

// ===== UPLL (UTMI) Hardware fixed at 480MHz.

// ===== USB Clock Source fixed at UPLL.

// ===== Target frequency (System clock)
// - XTAL frequency: 12MHz
// - System clock source: PLLA
// - System clock prescaler: 2 (divided by 2)
// - PLLA source: XTAL
// - PLLA output: XTAL * 16 / 1
// - System clock is: 12 * 16 / 1 / 2 = 96MHz
// ===== Target frequency (USB Clock)
// - USB clock source: UPLL
// - UPLL frequency: 480MHz
// - USB clock: 480MHz


#endif /* CONF_CLOCK_H_INCLUDED */
```

6.3.2.7 SAM3X, and SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock (MCK) Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_BYPASS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_4M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_8M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_12M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_BYPASS
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLLACK
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_UPLLCK

// ===== System Clock (MCK) Prescaler Options   (Fmck = Fsys / (SYSCLK_PRES))
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_1
#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_2
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_4
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_8
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_16
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_32
```

```
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_64
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_3

// ===== PLL0 (A) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
// Use mul and div effective values here.
#define CONFIG_PLL0_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL0_MUL             14
#define CONFIG_PLL0_DIV             1

// ===== UPLL (UTMI) Hardware fixed at 480MHz.

// ===== USB Clock Source Options   (Fusb = FpllX / USB_div)
// Use div effective value here.
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_UPLL
#define CONFIG_USBCLK_DIV           1

// ===== Target frequency (System clock)
// - XTAL frequency: 12MHz
// - System clock source: PLLA
// - System clock prescaler: 2 (divided by 2)
// - PLLA source: XTAL
// - PLLA output: XTAL * 14 / 1
// - System clock is: 12 * 14 / 1 /2 = 84MHz
// ===== Target frequency (USB Clock)
// - USB clock source: UPLL
// - USB clock divider: 1 (not divided)
// - UPLL frequency: 480MHz
// - USB clock: 480 / 1 = 480MHz


#endif /* CONF_CLOCK_H_INCLUDED */
```

6.3.3 conf_clocks.h

6.3.3.1 SAMD21 Device (USB)

```
#include <clock.h>

#ifndef CONF_CLOCKS_H_INCLUDED
#  define CONF_CLOCKS_H_INCLUDED

/* System clock bus configuration */
#  define CONF_CLOCK_CPU_CLOCK_FAILURE_DETECT     false
#  define CONF_CLOCK_FLASH_WAIT_STATES            2
#  define CONF_CLOCK_CPU_DIVIDER                  SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBA_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBB_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1

/* SYSTEM_CLOCK_SOURCE_OSC8M configuration - Internal 8MHz oscillator */
#  define CONF_CLOCK_OSC8M_PRESCALER              SYSTEM_OSC8M_DIV_1
#  define CONF_CLOCK_OSC8M_ON_DEMAND              true
#  define CONF_CLOCK_OSC8M_RUN_IN_STANDBY         false

/* SYSTEM_CLOCK_SOURCE_XOSC configuration - External clock/oscillator */
#  define CONF_CLOCK_XOSC_ENABLE                  false
#  define CONF_CLOCK_XOSC_EXTERNAL_CRYSTAL        SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC_EXTERNAL_FREQUENCY      12000000UL
#  define CONF_CLOCK_XOSC_STARTUP_TIME            SYSTEM_XOSC_STARTUP_32768
#  define CONF_CLOCK_XOSC_AUTO_GAIN_CONTROL       true
```

```
#  define CONF_CLOCK_XOSC_ON_DEMAND               true
#  define CONF_CLOCK_XOSC_RUN_IN_STANDBY          false

/* SYSTEM_CLOCK_SOURCE_XOSC32K configuration - External 32KHz crystal/clock oscillator */
#  define CONF_CLOCK_XOSC32K_ENABLE               false
#  define CONF_CLOCK_XOSC32K_EXTERNAL_CRYSTAL     SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC32K_STARTUP_TIME         SYSTEM_XOSC32K_STARTUP_65536
#  define CONF_CLOCK_XOSC32K_AUTO_AMPLITUDE_CONTROL  false
#  define CONF_CLOCK_XOSC32K_ENABLE_1KHZ_OUPUT    false
#  define CONF_CLOCK_XOSC32K_ENABLE_32KHZ_OUTPUT  true
#  define CONF_CLOCK_XOSC32K_ON_DEMAND            true
#  define CONF_CLOCK_XOSC32K_RUN_IN_STANDBY       false

/* SYSTEM_CLOCK_SOURCE_OSC32K configuration - Internal 32KHz oscillator */
#  define CONF_CLOCK_OSC32K_ENABLE                false
#  define CONF_CLOCK_OSC32K_STARTUP_TIME          SYSTEM_OSC32K_STARTUP_130
#  define CONF_CLOCK_OSC32K_ENABLE_1KHZ_OUTPUT    true
#  define CONF_CLOCK_OSC32K_ENABLE_32KHZ_OUTPUT   true
#  define CONF_CLOCK_OSC32K_ON_DEMAND             true
#  define CONF_CLOCK_OSC32K_RUN_IN_STANDBY        false

/* SYSTEM_CLOCK_SOURCE_DFLL configuration - Digital Frequency Locked Loop */
#  define CONF_CLOCK_DFLL_ENABLE                  true
#  define CONF_CLOCK_DFLL_LOOP_MODE               SYSTEM_CLOCK_DFLL_LOOP_MODE_USB_RECOVERY
#  define CONF_CLOCK_DFLL_ON_DEMAND               true

/* DFLL open loop mode configuration */
#  define CONF_CLOCK_DFLL_COARSE_VALUE            (0x1f / 4)
#  define CONF_CLOCK_DFLL_FINE_VALUE              (0xff / 4)

/* DFLL closed loop mode configuration */
#  define CONF_CLOCK_DFLL_SOURCE_GCLK_GENERATOR   GCLK_GENERATOR_1
#  define CONF_CLOCK_DFLL_MULTIPLY_FACTOR         (48000000 / 32768)
#  define CONF_CLOCK_DFLL_QUICK_LOCK              true
#  define CONF_CLOCK_DFLL_TRACK_AFTER_FINE_LOCK   true
#  define CONF_CLOCK_DFLL_KEEP_LOCK_ON_WAKEUP     true
#  define CONF_CLOCK_DFLL_ENABLE_CHILL_CYCLE      true
#  define CONF_CLOCK_DFLL_MAX_COARSE_STEP_SIZE    (0x1f / 4)
#  define CONF_CLOCK_DFLL_MAX_FINE_STEP_SIZE      (0xff / 4)

/* SYSTEM_CLOCK_SOURCE_DPLL configuration - Digital Phase-Locked Loop */
#  define CONF_CLOCK_DPLL_ENABLE                  false
#  define CONF_CLOCK_DPLL_ON_DEMAND               true
#  define CONF_CLOCK_DPLL_RUN_IN_STANDBY          false
#  define CONF_CLOCK_DPLL_LOCK_BYPASS             false
#  define CONF_CLOCK_DPLL_WAKE_UP_FAST            false
#  define CONF_CLOCK_DPLL_LOW_POWER_ENABLE        false

#  define CONF_CLOCK_DPLL_LOCK_TIME               SYSTEM_CLOCK_SOURCE_DPLL_LOCK_TIME_NO_TIMEOUT
#  define CONF_CLOCK_DPLL_REFERENCE_CLOCK         SYSTEM_CLOCK_SOURCE_DPLL_REFERENCE_CLOCK_REF0
#  define CONF_CLOCK_DPLL_FILTER                  SYSTEM_CLOCK_SOURCE_DPLL_FILTER_DEFAULT

#  define CONF_CLOCK_DPLL_REFERENCE_FREQUENCY     32768
#  define CONF_CLOCK_DPLL_REFEREMCE_DIVIDER       1
#  define CONF_CLOCK_DPLL_OUTPUT_FREQUENCY        48000000

/* Set this to true to configure the GCLK when running clocks_init. If set to
 * false, none of the GCLK generators will be configured in clocks_init(). */
#  define CONF_CLOCK_CONFIGURE_GCLK               true

/* Configure GCLK generator 0 (Main Clock) */
```

```
#  define CONF_CLOCK_GCLK_0_ENABLE                true
#  define CONF_CLOCK_GCLK_0_RUN_IN_STANDBY        true
#  define CONF_CLOCK_GCLK_0_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_DFLL
#  define CONF_CLOCK_GCLK_0_PRESCALER             1
#  define CONF_CLOCK_GCLK_0_OUTPUT_ENABLE         false

/* Configure GCLK generator 1 */
#  define CONF_CLOCK_GCLK_1_ENABLE                false
#  define CONF_CLOCK_GCLK_1_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_1_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_XOSC32K
#  define CONF_CLOCK_GCLK_1_PRESCALER             1
#  define CONF_CLOCK_GCLK_1_OUTPUT_ENABLE         false

/* Configure GCLK generator 2 (RTC) */
#  define CONF_CLOCK_GCLK_2_ENABLE                false
#  define CONF_CLOCK_GCLK_2_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_2_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC32K
#  define CONF_CLOCK_GCLK_2_PRESCALER             32
#  define CONF_CLOCK_GCLK_2_OUTPUT_ENABLE         false

/* Configure GCLK generator 3 */
#  define CONF_CLOCK_GCLK_3_ENABLE                false
#  define CONF_CLOCK_GCLK_3_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_3_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_3_PRESCALER             1
#  define CONF_CLOCK_GCLK_3_OUTPUT_ENABLE         false

/* Configure GCLK generator 4 */
#  define CONF_CLOCK_GCLK_4_ENABLE                false
#  define CONF_CLOCK_GCLK_4_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_4_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_4_PRESCALER             1
#  define CONF_CLOCK_GCLK_4_OUTPUT_ENABLE         false

/* Configure GCLK generator 5 */
#  define CONF_CLOCK_GCLK_5_ENABLE                false
#  define CONF_CLOCK_GCLK_5_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_5_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_5_PRESCALER             1
#  define CONF_CLOCK_GCLK_5_OUTPUT_ENABLE         false

/* Configure GCLK generator 6 */
#  define CONF_CLOCK_GCLK_6_ENABLE                false
#  define CONF_CLOCK_GCLK_6_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_6_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_6_PRESCALER             1
#  define CONF_CLOCK_GCLK_6_OUTPUT_ENABLE         false

/* Configure GCLK generator 7 */
#  define CONF_CLOCK_GCLK_7_ENABLE                false
#  define CONF_CLOCK_GCLK_7_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_7_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_7_PRESCALER             1
#  define CONF_CLOCK_GCLK_7_OUTPUT_ENABLE         false

#endif /* CONF_CLOCKS_H_INCLUDED */
```

6.3.4 conf_board.h

6.3.4.1 AT32UC3A0, AT32UC3A1, and AT32UC3B Devices (USBB)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

6.3.4.2 AT32UC3A3, and AT32UC3A4 Devices (USBB with High Speed Support)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

6.3.4.3 AT32UC3C, ATUCXXD, ATUCXXL3U, and ATUCXXL4U Devices (USBC)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

6.3.4.4 SAM3X, and SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// USB pins are used
#define CONF_BOARD_USB_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

6.3.4.5 SAMD21 Device (USB)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

/* Enable USB VBUS detect */
#define CONF_BOARD_USB_VBUS_DETECT

#endif /* CONF_BOARD_H_INCLUDED */
```

6.3.5 conf_access.h

6.3.5.1 AT32UC3A0, AT32UC3A1, and AT32UC3B Devices (USBB)

On EVK1100, the AT45DBx and one SD/MMC are for MSC.

```
#ifndef _CONF_ACCESS_H_
#define _CONF_ACCESS_H_

#include "compiler.h"
```

```
#include "board.h"


#define LUN_0                   DISABLE
#define LUN_1                   ENABLE
#define LUN_2                   ENABLE
#define LUN_3                   DISABLE
#define LUN_4                   DISABLE
#define LUN_5                   DISABLE
#define LUN_6                   DISABLE
#define LUN_7                   DISABLE
#define LUN_USB                 DISABLE


#define VIRTUAL_MEM                             LUN_0
#define LUN_ID_VIRTUAL_MEM                      LUN_ID_0
#define LUN_0_INCLUDE                           "virtual_mem.h"
#define Lun_0_test_unit_ready                   virtual_test_unit_ready
#define Lun_0_read_capacity                     virtual_read_capacity
#define Lun_0_wr_protect                        virtual_wr_protect
#define Lun_0_removal                           virtual_removal
#define Lun_0_usb_read_10                       virtual_usb_read_10
#define Lun_0_usb_write_10                      virtual_usb_write_10
#define Lun_0_mem_2_ram                         virtual_mem_2_ram
#define Lun_0_ram_2_mem                         virtual_ram_2_mem
#define LUN_0_NAME                              "\"On-Chip Virtual Memory\""

#define AT45DBX_MEM                             LUN_1
#define LUN_ID_AT45DBX_MEM                      LUN_ID_1
#define LUN_1_INCLUDE                           "at45dbx_mem.h"
#define Lun_1_test_unit_ready                   at45dbx_test_unit_ready
#define Lun_1_read_capacity                     at45dbx_read_capacity
#define Lun_1_wr_protect                        at45dbx_wr_protect
#define Lun_1_removal                           at45dbx_removal
#define Lun_1_usb_read_10                       at45dbx_usb_read_10
#define Lun_1_usb_write_10                      at45dbx_usb_write_10
#define Lun_1_mem_2_ram                         at45dbx_df_2_ram
#define Lun_1_ram_2_mem                         at45dbx_ram_2_df
#define LUN_1_NAME                              "\"AT45DBX Data Flash\""

#define SD_MMC_0_MEM                            LUN_2
#define LUN_ID_SD_MMC_0_MEM                     LUN_ID_2
#define LUN_2_INCLUDE                           "sd_mmc_mem.h"
#define Lun_2_test_unit_ready                   sd_mmc_test_unit_ready_0
#define Lun_2_read_capacity                     sd_mmc_read_capacity_0
#define Lun_2_wr_protect                        sd_mmc_wr_protect_0
#define Lun_2_removal                           sd_mmc_removal_0
#define Lun_2_usb_read_10                       sd_mmc_usb_read_10_0
#define Lun_2_usb_write_10                      sd_mmc_usb_write_10_0
#define Lun_2_mem_2_ram                         sd_mmc_mem_2_ram_0
#define Lun_2_ram_2_mem                         sd_mmc_ram_2_mem_0
#define LUN_2_NAME                              "\"SD/MMC Card Slot 0\""

#define MEM_USB                                 LUN_USB
#define LUN_ID_MEM_USB                          LUN_ID_USB
#define LUN_USB_INCLUDE                         "host_mem.h"
#define Lun_usb_test_unit_ready(lun)            host_test_unit_ready(lun)
#define Lun_usb_read_capacity(lun, nb_sect)     host_read_capacity(lun, nb_sect)
#define Lun_usb_read_sector_size(lun)           host_read_sector_size(lun)
#define Lun_usb_wr_protect(lun)                 host_wr_protect(lun)
#define Lun_usb_removal()                       host_removal()
```

```
#define Lun_usb_mem_2_ram(addr, ram)            host_read_10_ram(addr, ram)
#define Lun_usb_ram_2_mem(addr, ram)            host_write_10_ram(addr, ram)
#define LUN_USB_NAME                            "\"Host Mass-Storage Memory\""

#define memory_start_read_action(nb_sectors)    ui_start_read()
#define memory_stop_read_action()               ui_stop_read()
#define memory_start_write_action(nb_sectors)   ui_start_write()
#define memory_stop_write_action()              ui_stop_write()
#include "ui.h"

#define ACCESS_USB            true
#define ACCESS_MEM_TO_RAM     false
#define ACCESS_STREAM         false
#define ACCESS_STREAM_RECORD  false
#define ACCESS_MEM_TO_MEM     false
#define ACCESS_CODEC          false


#define GLOBAL_WR_PROTECT     false


#endif // _CONF_ACCESS_H_
```

6.3.5.2 AT32UC3A3, and AT32UC3A4 Devices (USBB with High Speed Support)

On EVK1104, the AT45DBx and two SD/MMC slots are for MSC.

```
#ifndef _CONF_ACCESS_H_
#define _CONF_ACCESS_H_

#include "compiler.h"
#include "board.h"


#define LUN_0                 DISABLE
#define LUN_1                 ENABLE
#define LUN_2                 ENABLE
#define LUN_3                 ENABLE
#define LUN_4                 DISABLE
#define LUN_5                 DISABLE
#define LUN_6                 DISABLE
#define LUN_7                 DISABLE
#define LUN_USB               DISABLE


#define VIRTUAL_MEM                             LUN_0
#define LUN_ID_VIRTUAL_MEM                      LUN_ID_0
#define LUN_0_INCLUDE                           "virtual_mem.h"
#define Lun_0_test_unit_ready                   virtual_test_unit_ready
#define Lun_0_read_capacity                     virtual_read_capacity
#define Lun_0_wr_protect                        virtual_wr_protect
#define Lun_0_removal                           virtual_removal
#define Lun_0_usb_read_10                       virtual_usb_read_10
#define Lun_0_usb_write_10                      virtual_usb_write_10
#define Lun_0_mem_2_ram                         virtual_mem_2_ram
#define Lun_0_ram_2_mem                         virtual_ram_2_mem
#define LUN_0_NAME                              "\"On-Chip Virtual Memory\""

#define AT45DBX_MEM                             LUN_1
#define LUN_ID_AT45DBX_MEM                      LUN_ID_1
#define LUN_1_INCLUDE                           "at45dbx_mem.h"
```

```
#define Lun_1_test_unit_ready                   at45dbx_test_unit_ready
#define Lun_1_read_capacity                     at45dbx_read_capacity
#define Lun_1_wr_protect                        at45dbx_wr_protect
#define Lun_1_removal                           at45dbx_removal
#define Lun_1_usb_read_10                       at45dbx_usb_read_10
#define Lun_1_usb_write_10                      at45dbx_usb_write_10
#define Lun_1_mem_2_ram                         at45dbx_df_2_ram
#define Lun_1_ram_2_mem                         at45dbx_ram_2_df
#define LUN_1_NAME                              "\"AT45DBX Data Flash\""

#define SD_MMC_0_MEM                            LUN_2
#define LUN_ID_SD_MMC_0_MEM                     LUN_ID_2
#define LUN_2_INCLUDE                           "sd_mmc_mem.h"
#define Lun_2_test_unit_ready                   sd_mmc_test_unit_ready_0
#define Lun_2_read_capacity                     sd_mmc_read_capacity_0
#define Lun_2_wr_protect                        sd_mmc_wr_protect_0
#define Lun_2_removal                           sd_mmc_removal_0
#define Lun_2_usb_read_10                       sd_mmc_usb_read_10_0
#define Lun_2_usb_write_10                      sd_mmc_usb_write_10_0
#define Lun_2_mem_2_ram                         sd_mmc_mem_2_ram_0
#define Lun_2_ram_2_mem                         sd_mmc_ram_2_mem_0
#define LUN_2_NAME                              "\"SD/MMC Card Slot 0\""

#define SD_MMC_1_MEM                            LUN_3
#define LUN_ID_SD_MMC_1_MEM                     LUN_ID_3
#define LUN_3_INCLUDE                           "sd_mmc_mem.h"
#define Lun_3_test_unit_ready                   sd_mmc_test_unit_ready_1
#define Lun_3_read_capacity                     sd_mmc_read_capacity_1
#define Lun_3_wr_protect                        sd_mmc_wr_protect_1
#define Lun_3_removal                           sd_mmc_removal_1
#define Lun_3_usb_read_10                       sd_mmc_usb_read_10_1
#define Lun_3_usb_write_10                      sd_mmc_usb_write_10_1
#define Lun_3_mem_2_ram                         sd_mmc_mem_2_ram_1
#define Lun_3_ram_2_mem                         sd_mmc_ram_2_mem_1
#define LUN_3_NAME                              "\"SD/MMC Card Slot 1\""

#define MEM_USB                                 LUN_USB
#define LUN_ID_MEM_USB                          LUN_ID_USB
#define LUN_USB_INCLUDE                         "host_mem.h"
#define Lun_usb_test_unit_ready(lun)            host_test_unit_ready(lun)
#define Lun_usb_read_capacity(lun, nb_sect)     host_read_capacity(lun, nb_sect)
#define Lun_usb_read_sector_size(lun)           host_read_sector_size(lun)
#define Lun_usb_wr_protect(lun)                 host_wr_protect(lun)
#define Lun_usb_removal()                       host_removal()
#define Lun_usb_mem_2_ram(addr, ram)            host_read_10_ram(addr, ram)
#define Lun_usb_ram_2_mem(addr, ram)            host_write_10_ram(addr, ram)
#define LUN_USB_NAME                            "\"Host Mass-Storage Memory\""

#define memory_start_read_action(nb_sectors)    ui_start_read()
#define memory_stop_read_action()               ui_stop_read()
#define memory_start_write_action(nb_sectors)   ui_start_write()
#define memory_stop_write_action()              ui_stop_write()
#include "ui.h"

#define ACCESS_USB              true
#define ACCESS_MEM_TO_RAM       false
#define ACCESS_STREAM           false
#define ACCESS_STREAM_RECORD    false
#define ACCESS_MEM_TO_MEM       false
#define ACCESS_CODEC            false
```

```
#define GLOBAL_WR_PROTECT    false


#endif // _CONF_ACCESS_H_
```

6.3.5.3 AT32UC3C, ATUCXXD, ATUCXXL3U, and ATUCXXL4U Devices (USBC)

On EVK1100, the AT45DBx and one SD/MMC are for MSC.

```
#ifndef _CONF_ACCESS_H_
#define _CONF_ACCESS_H_

#include "compiler.h"
#include "board.h"


#define LUN_0                DISABLE
#define LUN_1                ENABLE
#define LUN_2                ENABLE
#define LUN_3                DISABLE
#define LUN_4                DISABLE
#define LUN_5                DISABLE
#define LUN_6                DISABLE
#define LUN_7                DISABLE
#define LUN_USB              DISABLE


#define VIRTUAL_MEM                         LUN_0
#define LUN_ID_VIRTUAL_MEM                  LUN_ID_0
#define LUN_0_INCLUDE                       "virtual_mem.h"
#define Lun_0_test_unit_ready               virtual_test_unit_ready
#define Lun_0_read_capacity                 virtual_read_capacity
#define Lun_0_wr_protect                    virtual_wr_protect
#define Lun_0_removal                       virtual_removal
#define Lun_0_usb_read_10                   virtual_usb_read_10
#define Lun_0_usb_write_10                  virtual_usb_write_10
#define Lun_0_mem_2_ram                     virtual_mem_2_ram
#define Lun_0_ram_2_mem                     virtual_ram_2_mem
#define LUN_0_NAME                          "\"On-Chip Virtual Memory\""

#define AT45DBX_MEM                         LUN_1
#define LUN_ID_AT45DBX_MEM                  LUN_ID_1
#define LUN_1_INCLUDE                       "at45dbx_mem.h"
#define Lun_1_test_unit_ready               at45dbx_test_unit_ready
#define Lun_1_read_capacity                 at45dbx_read_capacity
#define Lun_1_wr_protect                    at45dbx_wr_protect
#define Lun_1_removal                       at45dbx_removal
#define Lun_1_usb_read_10                   at45dbx_usb_read_10
#define Lun_1_usb_write_10                  at45dbx_usb_write_10
#define Lun_1_mem_2_ram                     at45dbx_df_2_ram
#define Lun_1_ram_2_mem                     at45dbx_ram_2_df
#define LUN_1_NAME                          "\"AT45DBX Data Flash\""

#define SD_MMC_0_MEM                          LUN_2
#define LUN_ID_SD_MMC_0_MEM                   LUN_ID_2
#define LUN_2_INCLUDE                         "sd_mmc_mem.h"
#define Lun_2_test_unit_ready                 sd_mmc_test_unit_ready_0
#define Lun_2_read_capacity                   sd_mmc_read_capacity_0
#define Lun_2_wr_protect                      sd_mmc_wr_protect_0
#define Lun_2_removal                         sd_mmc_removal_0
```

```
#define Lun_2_usb_read_10                       sd_mmc_usb_read_10_0
#define Lun_2_usb_write_10                      sd_mmc_usb_write_10_0
#define Lun_2_mem_2_ram                         sd_mmc_mem_2_ram_0
#define Lun_2_ram_2_mem                         sd_mmc_ram_2_mem_0
#define LUN_2_NAME                              "\"SD/MMC Card Slot 0\""

#define MEM_USB                                 LUN_USB
#define LUN_ID_MEM_USB                          LUN_ID_USB
#define LUN_USB_INCLUDE                         "host_mem.h"
#define Lun_usb_test_unit_ready(lun)            host_test_unit_ready(lun)
#define Lun_usb_read_capacity(lun, nb_sect)     host_read_capacity(lun, nb_sect)
#define Lun_usb_read_sector_size(lun)           host_read_sector_size(lun)
#define Lun_usb_wr_protect(lun)                 host_wr_protect(lun)
#define Lun_usb_removal()                       host_removal()
#define Lun_usb_mem_2_ram(addr, ram)            host_read_10_ram(addr, ram)
#define Lun_usb_ram_2_mem(addr, ram)            host_write_10_ram(addr, ram)
#define LUN_USB_NAME                            "\"Host Mass-Storage Memory\""

#define memory_start_read_action(nb_sectors)    ui_start_read()
#define memory_stop_read_action()               ui_stop_read()
#define memory_start_write_action(nb_sectors)   ui_start_write()
#define memory_stop_write_action()              ui_stop_write()
#include "ui.h"

#define ACCESS_USB           true
#define ACCESS_MEM_TO_RAM    false
#define ACCESS_STREAM        false
#define ACCESS_STREAM_RECORD false
#define ACCESS_MEM_TO_MEM    false
#define ACCESS_CODEC         false


#define GLOBAL_WR_PROTECT    false


#endif // _CONF_ACCESS_H_
```

6.3.5.4 SAM3X, and SAM3A Devices (UOTGHS: USB OTG High Speed)

On SAM3X-EK, the SD/MMC and on-board nand are for MSC.

```
#ifndef _CONF_ACCESS_H_
#define _CONF_ACCESS_H_

#include "compiler.h"
#include "board.h"


#define LUN_0               DISABLE
#define LUN_1               DISABLE
#define LUN_2               ENABLE
#define LUN_3               DISABLE
#define LUN_4               ENABLE
#define LUN_5               DISABLE
#define LUN_6               DISABLE
#define LUN_7               DISABLE
#define LUN_USB             DISABLE


#define VIRTUAL_MEM                             LUN_0
#define LUN_ID_VIRTUAL_MEM                      LUN_ID_0
```

```
#define LUN_0_INCLUDE                       "virtual_mem.h"
#define Lun_0_test_unit_ready               virtual_test_unit_ready
#define Lun_0_read_capacity                 virtual_read_capacity
#define Lun_0_unload                        NULL /* Can not be unloaded */
#define Lun_0_wr_protect                    virtual_wr_protect
#define Lun_0_removal                       virtual_removal
#define Lun_0_usb_read_10                   virtual_usb_read_10
#define Lun_0_usb_write_10                  virtual_usb_write_10
#define Lun_0_mem_2_ram                     virtual_mem_2_ram
#define Lun_0_ram_2_mem                     virtual_ram_2_mem
#define LUN_0_NAME                          "\"On-Chip Virtual Memory\""

#define AT45DBX_MEM                         LUN_1
#define LUN_ID_AT45DBX_MEM                  LUN_ID_1
#define LUN_1_INCLUDE                       "at45dbx_mem.h"
#define Lun_1_test_unit_ready               at45dbx_test_unit_ready
#define Lun_1_read_capacity                 at45dbx_read_capacity
#define Lun_1_unload                        NULL /* Can not be unloaded */
#define Lun_1_wr_protect                    at45dbx_wr_protect
#define Lun_1_removal                       at45dbx_removal
#define Lun_1_usb_read_10                   at45dbx_usb_read_10
#define Lun_1_usb_write_10                  at45dbx_usb_write_10
#define Lun_1_mem_2_ram                     at45dbx_df_2_ram
#define Lun_1_ram_2_mem                     at45dbx_ram_2_df
#define LUN_1_NAME                          "\"AT45DBX Data Flash\""

#define SD_MMC_0_MEM                        LUN_2
#define LUN_ID_SD_MMC_0_MEM                 LUN_ID_2
#define LUN_2_INCLUDE                       "sd_mmc_mem.h"
#define Lun_2_test_unit_ready               sd_mmc_test_unit_ready_0
#define Lun_2_read_capacity                 sd_mmc_read_capacity_0
#define Lun_2_unload                        sd_mmc_unload_0
#define Lun_2_wr_protect                    sd_mmc_wr_protect_0
#define Lun_2_removal                       sd_mmc_removal_0
#define Lun_2_usb_read_10                   sd_mmc_usb_read_10_0
#define Lun_2_usb_write_10                  sd_mmc_usb_write_10_0
#define Lun_2_mem_2_ram                     sd_mmc_mem_2_ram_0
#define Lun_2_ram_2_mem                     sd_mmc_ram_2_mem_0
#define LUN_2_NAME                          "\"SD/MMC Card Slot 0\""

#define NAND_FLASH_MEM                      LUN_4
#define LUN_ID_NAND_FLASH_MEM               LUN_ID_4
#define LUN_4_INCLUDE                       "nand_flash_mem.h"
#define Lun_4_test_unit_ready               nand_flash_test_unit_ready
#define Lun_4_read_capacity                 nand_flash_read_capacity
#define Lun_4_unload                        NULL
#define Lun_4_wr_protect                    nand_flash_wr_protect
#define Lun_4_removal                       nand_flash_removal
#define Lun_4_usb_read_10                   nand_flash_usb_read_10
#define Lun_4_usb_write_10                  nand_flash_usb_write_10
#define Lun_4_mem_2_ram                     nand_flash_mem_2_ram
#define Lun_4_ram_2_mem                     nand_flash_ram_2_mem
#define LUN_4_NAME                          "\"nand_flash on EBI\""

#define MEM_USB                             LUN_USB
#define LUN_ID_MEM_USB                      LUN_ID_USB
#define LUN_USB_INCLUDE                     "uhi_msc_mem.h"
#define Lun_usb_get_lun()                   uhi_msc_mem_get_lun()
#define Lun_usb_test_unit_ready(lun)        uhi_msc_mem_test_unit_ready(lun)
#define Lun_usb_read_capacity(lun, nb_sect) uhi_msc_mem_read_capacity(lun, nb_sect)
#define Lun_usb_read_sector_size(lun)       uhi_msc_mem_read_sector_size(lun)
```

```
#define Lun_usb_wr_protect(lun)                 uhi_msc_mem_wr_protect(lun)
#define Lun_usb_removal()                       uhi_msc_mem_removal()
#define Lun_usb_mem_2_ram(addr, ram)            uhi_msc_mem_read_10_ram(addr, ram)
#define Lun_usb_ram_2_mem(addr, ram)            uhi_msc_mem_write_10_ram(addr, ram)
#define LUN_USB_NAME                            "\"Host Mass-Storage Memory\""

#define memory_start_read_action(nb_sectors)    ui_start_read()
#define memory_stop_read_action()               ui_stop_read()
#define memory_start_write_action(nb_sectors)   ui_start_write()
#define memory_stop_write_action()              ui_stop_write()
#include "ui.h"

#define ACCESS_USB           true
#define ACCESS_MEM_TO_RAM    false
#define ACCESS_STREAM        false
#define ACCESS_STREAM_RECORD false
#define ACCESS_MEM_TO_MEM    false
#define ACCESS_CODEC         false


#define GLOBAL_WR_PROTECT    false


#endif // _CONF_ACCESS_H_
```

6.3.5.5 SAMD21 Device (USB)

```
#ifndef _CONF_ACCESS_H_
#define _CONF_ACCESS_H_

#include "compiler.h"
#include "board.h"


#define LUN_0                ENABLE
#define LUN_1                DISABLE
#define LUN_2                DISABLE
#define LUN_3                DISABLE
#define LUN_4                DISABLE
#define LUN_5                DISABLE
#define LUN_6                DISABLE
#define LUN_7                DISABLE
#define LUN_USB              DISABLE


#define VIRTUAL_MEM                             LUN_0
#define LUN_ID_VIRTUAL_MEM                      LUN_ID_0
#define LUN_0_INCLUDE                           "virtual_mem.h"
#define Lun_0_test_unit_ready                   virtual_test_unit_ready
#define Lun_0_read_capacity                     virtual_read_capacity
#define Lun_0_unload                            NULL /* Can not be unloaded */
#define Lun_0_wr_protect                        virtual_wr_protect
#define Lun_0_removal                           virtual_removal
#define Lun_0_usb_read_10                       virtual_usb_read_10
#define Lun_0_usb_write_10                      virtual_usb_write_10
#define Lun_0_mem_2_ram                         virtual_mem_2_ram
#define Lun_0_ram_2_mem                         virtual_ram_2_mem
#define LUN_0_NAME                              "\"On-Chip Virtual Memory\""

#define MEM_USB                                 LUN_USB
```

```
#define LUN_ID_MEM_USB                          LUN_ID_USB
#define LUN_USB_INCLUDE                         "uhi_msc_mem.h"
#define Lun_usb_get_lun()                       uhi_msc_mem_get_lun()
#define Lun_usb_test_unit_ready(lun)            uhi_msc_mem_test_unit_ready(lun)
#define Lun_usb_read_capacity(lun, nb_sect)     uhi_msc_mem_read_capacity(lun, nb_sect)
#define Lun_usb_read_sector_size(lun)           uhi_msc_mem_read_sector_size(lun)
#define Lun_usb_wr_protect(lun)                 uhi_msc_mem_wr_protect(lun)
#define Lun_usb_removal()                       uhi_msc_mem_removal()
#define Lun_usb_mem_2_ram(addr, ram)            uhi_msc_mem_read_10_ram(addr, ram)
#define Lun_usb_ram_2_mem(addr, ram)            uhi_msc_mem_write_10_ram(addr, ram)
#define LUN_USB_NAME                            "\"Host Mass-Storage Memory\""

#define memory_start_read_action(nb_sectors)    ui_start_read()
#define memory_stop_read_action()               ui_stop_read()
#define memory_start_write_action(nb_sectors)   ui_start_write()
#define memory_stop_write_action()              ui_stop_write()
#include "ui.h"

#define ACCESS_USB           true
#define ACCESS_MEM_TO_RAM    false
#define ACCESS_STREAM        false
#define ACCESS_STREAM_RECORD false
#define ACCESS_MEM_TO_MEM    false
#define ACCESS_CODEC         false


#define GLOBAL_WR_PROTECT    false


#endif // _CONF_ACCESS_H_
```

6.3.6 conf_virtual_mem.h

6.3.6.1 On-chip Virtual Memory Disk

```
#ifndef _CONF_VIRTUAL_MEM_H_
#define _CONF_VIRTUAL_MEM_H_

#define VMEM_NB_SECTOR 48 // Internal RAM 24KB (should > 20KB or PC can not format it)

#endif // _CONF_VIRTUAL_MEM_H_
```

6.3.6.2 On-board Virtual Memory Disk

```
#ifndef _CONF_VIRTUAL_MEM_H_
#define _CONF_VIRTUAL_MEM_H_

// Start address of Virtual Memory if it's on external RAM (PSRAM)
#define VMEM_ADDRESS   0x60000000
#define VMEM_NB_SECTOR 2048 // External PSRAM 1MB

#endif // _CONF_VIRTUAL_MEM_H_
```

7. USB Device Interface (UDI) for Vendor Class Device

USB Device Interface (UDI) for Vendor Class Device provides an interface for the configuration and management of USB Vendor Device.

The outline of this documentation is as follows:

- API Overview
- Quick Start Guide for USB Device Vendor Module (UDI Vendor)
- Configuration File Examples

For more details for Atmel® Software Framework (ASF) USB Device Stack, and USB Device Vendor, refer to following application notes:

- AVR4900: ASF - USB Device Stack[1]
- AVR4901: ASF - USB Device Vendor Class Application[2].
- AVR4920: ASF - USB Device Stack - Compliance and Performance Figures[3]
- AVR4921: ASF - USB Device Stack Differences between ASF V1 and V2[4]

7.1 API Overview

7.1.1 Variable and Type Definitions

7.1.1.1 Interface with USB Device Core (UDC)

Variable required by UDC.

Variable udi_api_vendor

```
UDC_DESC_STORAGE udi_api_t udi_api_vendor
```

Global structure which contains standard UDI interface for UDC.

7.1.2 Structure Definitions

7.1.2.1 Struct udi_vendor_desc_t

Interface descriptor structure for vendor Class interface.

Table 7-1. Members

Type	Name	Description
usb_iface_desc_t	iface0	Standard USB interface descriptor structure
usb_iface_desc_t	iface1	Standard USB interface descriptor structure

[1] http://www.atmel.com/dyn/resources/prod_documents/doc8360.pdf
[2] http://www.atmel.com/dyn/resources/prod_documents/doc8481.pdf
[3] http://www.atmel.com/dyn/resources/prod_documents/doc8410.pdf
[4] http://www.atmel.com/dyn/resources/prod_documents/doc8411.pdf

7.1.3 Macro Definitions

7.1.3.1 USB Interface Descriptors

The following structures provide predefined USB interface descriptors. It must be used to define the final USB descriptors.

Macro UDI_VENDOR_EPS_INT_DESC

```
#define UDI_VENDOR_EPS_INT_DESC
```

Endpoint descriptors.

Macro UDI_VENDOR_EPS_INT_DESC_FS

```
#define UDI_VENDOR_EPS_INT_DESC_FS
```

Macro UDI_VENDOR_EPS_INT_DESC_HS

```
#define UDI_VENDOR_EPS_INT_DESC_HS
```

Macro UDI_VENDOR_EPS_BULK_DESC

```
#define UDI_VENDOR_EPS_BULK_DESC
```

Macro UDI_VENDOR_EPS_BULK_DESC_FS

```
#define UDI_VENDOR_EPS_BULK_DESC_FS
```

Macro UDI_VENDOR_EPS_BULK_DESC_HS

```
#define UDI_VENDOR_EPS_BULK_DESC_HS
```

Macro UDI_VENDOR_EPS_ISO_DESC

```
#define UDI_VENDOR_EPS_ISO_DESC
```

Macro UDI_VENDOR_EPS_ISO_DESC_FS

```
#define UDI_VENDOR_EPS_ISO_DESC_FS
```

Macro UDI_VENDOR_EPS_ISO_DESC_HS

```
#define UDI_VENDOR_EPS_ISO_DESC_HS
```

Macro UDI_VENDOR_STRING_ID

```
#define UDI_VENDOR_STRING_ID 0
```

By default no string is associated to this interface.

Macro UDI_VENDOR_EP_NB_INT

```
#define UDI_VENDOR_EP_NB_INT ((UDI_VENDOR_EPS_SIZE_INT_FS)?2:0)
```

Maximum six endpoints used by vendor interface.

Macro UDI_VENDOR_EP_NB_BULK

```
#define UDI_VENDOR_EP_NB_BULK ((UDI_VENDOR_EPS_SIZE_BULK_FS)?2:0)
```

Macro UDI_VENDOR_EP_NB_ISO

```
#define UDI_VENDOR_EP_NB_ISO ((UDI_VENDOR_EPS_SIZE_ISO_FS)?2:0)
```

Macro UDI_VENDOR_EP_NB

```
#define UDI_VENDOR_EP_NB \
  (UDI_VENDOR_EP_NB_INT+UDI_VENDOR_EP_NB_BULK+UDI_VENDOR_EP_NB_ISO)
```

Macro UDI_VENDOR_DESC

```
#define UDI_VENDOR_DESC \
  .iface0.bLength            = sizeof(usb_iface_desc_t),\
 .iface0.bDescriptorType    = USB_DT_INTERFACE,\
 .iface0.bInterfaceNumber   = UDI_VENDOR_IFACE_NUMBER,\
```

```
  .iface0.bAlternateSetting  = 0,\
  .iface0.bNumEndpoints      = 0,\
  .iface0.bInterfaceClass    = VENDOR_CLASS,\
  .iface0.bInterfaceSubClass = VENDOR_SUBCLASS,\
  .iface0.bInterfaceProtocol = VENDOR_PROTOCOL,\
  .iface0.iInterface         = UDI_VENDOR_STRING_ID,\
  .iface1.bLength            = sizeof(usb_iface_desc_t),\
  .iface1.bDescriptorType    = USB_DT_INTERFACE,\
  .iface1.bInterfaceNumber   = UDI_VENDOR_IFACE_NUMBER,\
  .iface1.bAlternateSetting  = 1,\
  .iface1.bNumEndpoints      = UDI_VENDOR_EP_NB,\
  .iface1.bInterfaceClass    = VENDOR_CLASS,\
  .iface1.bInterfaceSubClass = VENDOR_SUBCLASS,\
  .iface1.bInterfaceProtocol = VENDOR_PROTOCOL,\
  .iface1.iInterface         = UDI_VENDOR_STRING_ID,\
 UDI_VENDOR_EPS_INT_DESC \
 UDI_VENDOR_EPS_BULK_DESC \
 UDI_VENDOR_EPS_ISO_DESC \
```

Content of vendor interface descriptor for all speeds.

Macro UDI_VENDOR_DESC_FS

```
#define UDI_VENDOR_DESC_FS \
  {\
 UDI_VENDOR_DESC \
 UDI_VENDOR_EPS_INT_DESC_FS \
 UDI_VENDOR_EPS_BULK_DESC_FS \
 UDI_VENDOR_EPS_ISO_DESC_FS \
   }
```

Content of vendor interface descriptor for full speed only.

Macro UDI_VENDOR_DESC_HS

```
#define UDI_VENDOR_DESC_HS \
  {\
 UDI_VENDOR_DESC \
 UDI_VENDOR_EPS_INT_DESC_HS \
 UDI_VENDOR_EPS_BULK_DESC_HS \
 UDI_VENDOR_EPS_ISO_DESC_HS \
   }
```

Content of vendor interface descriptor for high speed only.

7.1.4 Function Definitions

7.1.4.1 USB Device Interface (UDI) for Vendor Class

Common APIs used by high level application to use this USB class.

These routines are used to transfer data to/from USB VENDOR endpoints.

See Quick start guide for USB Device vendor module.

Function udi_vendor_interrupt_in_run()

Start a transfer on interrupt IN.

```
bool udi_vendor_interrupt_in_run(
  uint8_t * buf,
  iram_size_t buf_size,
  udd_callback_trans_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 7-2. Parameters

Data direction	Parameter name	Description
[in]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns 1 if function was successfully done, otherwise 0.

Function udi_vendor_interrupt_out_run()

Start a transfer on interrupt OUT.

```
bool udi_vendor_interrupt_out_run(
  uint8_t * buf,
  iram_size_t buf_size,
  udd_callback_trans_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 7-3. Parameters

Data direction	Parameter name	Description
[in]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns 1 if function was successfully done, otherwise 0.

Function udi_vendor_bulk_in_run()

Start a transfer on bulk IN.

```
bool udi_vendor_bulk_in_run(
  uint8_t * buf,
  iram_size_t buf_size,
  udd_callback_trans_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 7-4. Parameters

Data direction	Parameter name	Description
[in]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns 1 if function was successfully done, otherwise 0.

Function udi_vendor_bulk_out_run()

Start a transfer on bulk OUT.

```
bool udi_vendor_bulk_out_run(
  uint8_t * buf,
  iram_size_t buf_size,
  udd_callback_trans_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 7-5. Parameters

Data direction	Parameter name	Description
[in]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns 1 if function was successfully done, otherwise 0.

Function udi_vendor_iso_in_run()

Start a transfer on isochronous IN.

```
bool udi_vendor_iso_in_run(
  uint8_t * buf,
  iram_size_t buf_size,
  udd_callback_trans_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 7-6. Parameters

Data direction	Parameter name	Description
[in]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns 1 if function was successfully done, otherwise 0.

Function udi_vendor_iso_out_run()

Start a transfer on isochronous OUT.

```
bool udi_vendor_iso_out_run(
  uint8_t * buf,
  iram_size_t buf_size,
  udd_callback_trans_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 7-7. Parameters

Data direction	Parameter name	Description
[in]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns 1 if function was successfully done, otherwise 0.

7.2 Quick Start Guide for USB Device Vendor Module (UDI Vendor)

This is the quick start guide for the USB Device Vendor Module (UDI Vendor) with step-by-step instructions on how to configure and use the modules in a selection of use cases.

The use cases highlights several code fragments. The code fragments in the steps for setup can be copied into a custom initialization function, while the steps for usage can be copied into, e.g., the main application function.

7.2.1 Basic Use Case

In this basic use case, the "USB Vendor (Single Class support)" module is used. The "USB Vendor (Composite Device)" module usage is described in Advanced Use Cases.

7.2.1.1 Setup Steps

As a USB device, it follows common USB device setup steps. Refer to USB Device Basic Setup.

7.2.1.2 Usage Steps

Example code

Content of conf_usb.h:

```
#define UDI_VENDOR_ENABLE_EXT() my_callback_vendor_enable()
extern bool my_callback_vendor_enable(void);
#define UDI_VENDOR_DISABLE_EXT() my_callback_vendor_disable()
extern void my_callback_vendor_disable(void);

#define  UDI_VENDOR_SETUP_OUT_RECEIVED()  my_vendor_setup_out_received()
extern bool my_vendor_setup_out_received(void);
#define  UDI_VENDOR_SETUP_IN_RECEIVED()   my_vendor_setup_in_received()
extern bool my_vendor_setup_in_received(void);

#define  UDI_VENDOR_EPS_SIZE_INT_FS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_FS   64
#define  UDI_VENDOR_EPS_SIZE_ISO_FS   256
#define  UDI_VENDOR_EPS_SIZE_INT_HS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_HS  512
#define  UDI_VENDOR_EPS_SIZE_ISO_HS    64

#include "udi_vendor_conf.h" // At the end of conf_usb.h file
```

Add to application C-file:

```
static bool my_flag_autorize_vendor_transfert = false;
bool my_callback_vendor_enable(void)
{
   my_flag_autorize_vendor_transfert = true;
   return true;
}
void my_callback_vendor_disable(void)
{
   my_flag_autorize_vendor_transfert = false;
}

uint8_t global_buffer[X];
void task(void)
{
   if (my_flag_autorize_vendor_transfert) {
     // Enable a transfer on OUT interrupt endpoint
     udi_vendor_interrupt_out_run(
            global_buffer,
            sizeof(global_buffer),
            NULL);
     // Enable a transfer on IN interrupt endpoint
     udi_vendor_interrupt_in_run(
```

```
            global_buffer,
            sizeof(global_buffer),
            NULL);
    ...
  }
}
```

7.2.1.3 Workflow

1. Ensure that conf_usb.h is available and contains the following configuration, which is the USB device Vendor configuration:

```
#define UDI_VENDOR_ENABLE_EXT() my_callback_vendor_enable()
extern bool my_callback_vendor_enable(void);
```

Note After the device enumeration (detecting and identifying USB devices), the USB host starts the device configuration. When the USB Vendor interface from the device is accepted by the host, the USB host enables this interface and the UDI_VENDOR_ENABLE_EXT() callback function is called and return true. Thus, when this event is received, the Vendor transfers can start.

```
#define UDI_VENDOR_DISABLE_EXT() my_callback_vendor_disable()
extern void my_callback_vendor_disable(void);
```

Note When the USB device is unplugged or is reset by the USB host, the USB interface is disabled and the UDI_VENDOR_DISABLE_EXT() callback function is called. Thus, it is recommended to disable the data Vendor transfer.

```
#define  UDI_VENDOR_SETUP_OUT_RECEIVED()  my_vendor_setup_out_received()
extern bool my_vendor_setup_out_received(void);
#define  UDI_VENDOR_SETUP_IN_RECEIVED()   my_vendor_setup_in_received()
extern bool my_vendor_setup_in_received(void);
```

Note The control requests for the interface Vendor will be processed through these both callbacks.

```
#define  UDI_VENDOR_EPS_SIZE_INT_FS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_FS   64
#define  UDI_VENDOR_EPS_SIZE_ISO_FS   256
#define  UDI_VENDOR_EPS_SIZE_INT_HS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_HS  512
#define  UDI_VENDOR_EPS_SIZE_ISO_HS    64
```

Note The endpoint size is defined by the final application, and can be disabled if the full speed size is zero.

2. The Vendor transfers on interrupt, bulk, and isochronous endpoints are done through these functions:

```
// Start a transfer on interrupt IN
udi_vendor_interrupt_in_run();
```

```
// Start a transfer on interrupt OUT
udi_vendor_interrupt_out_run();
// Start a transfer on bulk IN
udi_vendor_bulk_in_run();
// Start a transfer on bulk OUT
udi_vendor_bulk_out_run();
// Start a transfer on isochronous IN
udi_vendor_iso_in_run();
// Start a transfer on isochronous OUT
udi_vendor_iso_out_run();
```

7.2.2 Advanced Use Cases

For multiple interface use of UDI Vendor module, see the following:

- Vendor in a Composite Device

For more advanced use of the UDI Vendor module, see the following:

- USB Device Advanced Use Cases

7.2.3 Vendor in a Composite Device

A USB Composite Device is a USB Device, which uses more than one USB class. In this use case, the "USB Vendor (Composite Device)" module is used to create a USB composite device. Thus, this USB module can be associated with another "Composite Device" module, like "USB HID Mouse (Composite Device)".

Also, you can refer to application note AVR4902 ASF - USB Composite Device[5].

7.2.3.1 Setup Steps

For the setup code of this use case to work, the Basic Use Case must be followed.

7.2.3.2 Usage Steps

Example Code

Content of conf_usb.h:

```
#define USB_DEVICE_EP_CTRL_SIZE  64
#define USB_DEVICE_NB_INTERFACE (X+1)
#define USB_DEVICE_MAX_EP (X) to (X+6)

#define  UDI_VENDOR_EP_INTERRUPT_IN  (1 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_INTERRUPT_OUT (2 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_BULK_IN       (3 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_BULK_OUT      (4 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_ISO_IN        (5 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_ISO_OUT       (6 | USB_EP_DIR_OUT)

#define UDI_VENDOR_IFACE_NUMBER  X

#define UDI_COMPOSITE_DESC_T \
   udi_vendor_desc_t udi_vendor; \
   ...
#define UDI_COMPOSITE_DESC_FS \
   .udi_vendor = UDI_VENDOR_DESC, \
   ...
#define UDI_COMPOSITE_DESC_HS \
   .udi_vendor = UDI_VENDOR_DESC, \
```

[5] http://www.atmel.com/dyn/resources/prod_documents/doc8445.pdf

```
   ...
#define UDI_COMPOSITE_API \
   &udi_api_vendor, \
   ...
```

Workflow

1. Ensure that conf_usb.h is available and contains the following parameters required for a USB composite device configuration:

```
// Endpoint control size, This must be:
// - 8, 16, 32 or 64 for full speed device (8 is recommended to save RAM)
// - 64 for a high speed device
#define USB_DEVICE_EP_CTRL_SIZE  64
// Total Number of interfaces on this USB device.
// Add 1 for Vendor.
#define USB_DEVICE_NB_INTERFACE (X+1)
// Total number of endpoints on this USB device.
// This must include each endpoint for each interface.
// Add 0 to 6 for Vendor interface.
// The number depends on UDI_VENDOR_EPS_SIZE_..._FS defines.
#define USB_DEVICE_MAX_EP (X) to (X+6)
```

2. Ensure that conf_usb.h contains the description of composite device:

```
// The endpoint numbers chosen by you for the Vendor.
// The endpoint numbers starting from 1.
#define  UDI_VENDOR_EP_INTERRUPT_IN  (1 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_INTERRUPT_OUT (2 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_BULK_IN       (3 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_BULK_OUT      (4 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_ISO_IN        (5 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_ISO_OUT       (6 | USB_EP_DIR_OUT)
// The interface index of an interface starting from 0
#define UDI_VENDOR_IFACE_NUMBER  X
```

3. Ensure that conf_usb.h contains the following parameters required for a USB composite device configuration:

```
// USB Interfaces descriptor structure
#define UDI_COMPOSITE_DESC_T \
   ...
   udi_vendor_desc_t udi_vendor; \
   ...
// USB Interfaces descriptor value for Full Speed
#define UDI_COMPOSITE_DESC_FS \
   ...
   .udi_vendor = UDI_VENDOR_DESC_FS, \
   ...
// USB Interfaces descriptor value for High Speed
#define UDI_COMPOSITE_DESC_HS \
   ...
   .udi_vendor = UDI_VENDOR_DESC_HS, \
   ...
// USB Interface APIs
#define UDI_COMPOSITE_API \
   ...
   &udi_api_vendor, \
   ...
```

Note The descriptors order given in the four lists above must be the same as the order defined by all interface indexes. The interface index orders are defined through UDI_X_IFACE_NUMBER defines.

7.3 Configuration File Examples

7.3.1 conf_usb.h

7.3.1.1 UDI Vendor Single

```
#ifndef _CONF_USB_H_
#define _CONF_USB_H_

#include "compiler.h"

#warning You must refill the following definitions with a correct values

#define  USB_DEVICE_VENDOR_ID             USB_VID_ATMEL
#define  USB_DEVICE_PRODUCT_ID            USB_PID_ATMEL_ASF_HIDGENERIC
#define  USB_DEVICE_MAJOR_VERSION         1
#define  USB_DEVICE_MINOR_VERSION         0
#define  USB_DEVICE_POWER                 100 // Consumption on Vbus line (mA)
#define  USB_DEVICE_ATTR                  \
    (USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_BUS_POWERED)
//  (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_SELF_POWERED)
//  (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_BUS_POWERED)

// #define  USB_DEVICE_MANUFACTURE_NAME      "Manufacture name"
// #define  USB_DEVICE_PRODUCT_NAME          "Product name"
// #define  USB_DEVICE_SERIAL_NAME           "12...EF"

//#define  USB_DEVICE_LOW_SPEED

#if (UC3A3||UC3A4)
//#define  USB_DEVICE_HS_SUPPORT
#endif


// #define  UDC_VBUS_EVENT(b_vbus_high)      user_callback_vbus_action(b_vbus_high)
// extern void user_callback_vbus_action(bool b_vbus_high);
// #define  UDC_SOF_EVENT()                  user_callback_sof_action()
// extern void user_callback_sof_action(void);
// #define  UDC_SUSPEND_EVENT()              user_callback_suspend_action()
// extern void user_callback_suspend_action(void);
// #define  UDC_RESUME_EVENT()               user_callback_resume_action()
// extern void user_callback_resume_action(void);
// #define  UDC_REMOTEWAKEUP_ENABLE()        user_callback_remotewakeup_enable()
// extern void user_callback_remotewakeup_enable(void);
// #define  UDC_REMOTEWAKEUP_DISABLE()       user_callback_remotewakeup_disable()
// extern void user_callback_remotewakeup_disable(void);
// #define  UDC_GET_EXTRA_STRING()



#define  UDI_HID_GENERIC_ENABLE_EXT()       true
#define  UDI_HID_GENERIC_DISABLE_EXT()
#define  UDI_HID_GENERIC_REPORT_OUT(ptr)
```

```
#define  UDI_HID_GENERIC_SET_FEATURE(f)
/*
 * #define UDI_HID_GENERIC_ENABLE_EXT() my_callback_generic_enable()
 * extern bool my_callback_generic_enable(void);
 * #define UDI_HID_GENERIC_DISABLE_EXT() my_callback_generic_disable()
 * extern void my_callback_generic_disable(void);
 * #define  UDI_HID_GENERIC_REPORT_OUT(ptr) my_callback_generic_report_out(ptr)
 * extern void my_callback_generic_report_out(uint8_t *report);
 * #define  UDI_HID_GENERIC_SET_FEATURE(f) my_callback_generic_set_feature(f)
 * extern void my_callback_generic_set_feature(uint8_t *report_feature);
 */

#define  UDI_HID_REPORT_IN_SIZE             64
#define  UDI_HID_REPORT_OUT_SIZE            64
#define  UDI_HID_REPORT_FEATURE_SIZE        4

#define  UDI_HID_GENERIC_EP_SIZE            64




#include "udi_hid_generic_conf.h"

#endif // _CONF_USB_H_
```

7.3.1.2 UDI Vendor Multiple (Composite)

```
#ifndef _CONF_USB_H_
#define _CONF_USB_H_

#include "compiler.h"

#warning You must refill the following definitions with a correct values

#define  USB_DEVICE_VENDOR_ID             USB_VID_ATMEL
#define  USB_DEVICE_PRODUCT_ID            0xFFFF
#define  USB_DEVICE_MAJOR_VERSION         1
#define  USB_DEVICE_MINOR_VERSION         0
#define  USB_DEVICE_POWER                 100 // Consumption on VBUS line (mA)
#define  USB_DEVICE_ATTR                  \
        (USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_BUS_POWERED)
// (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_SELF_POWERED)
// (USB_CONFIG_ATTR_REMOTE_WAKEUP|USB_CONFIG_ATTR_BUS_POWERED)

// #define  USB_DEVICE_MANUFACTURE_NAME      "Manufacture name"
// #define  USB_DEVICE_PRODUCT_NAME          "Product name"
// #define  USB_DEVICE_SERIAL_NAME           "12...EF" // Disk SN for MSC

//#define  USB_DEVICE_LOW_SPEED

#if (UC3A3||UC3A4)
//#define  USB_DEVICE_HS_SUPPORT
#endif


// #define  UDC_VBUS_EVENT(b_vbus_high)      user_callback_vbus_action(b_vbus_high)
// extern void user_callback_vbus_action(bool b_vbus_high);
// #define  UDC_SOF_EVENT()                  user_callback_sof_action()
```

```
// extern void user_callback_sof_action(void);
// #define  UDC_SUSPEND_EVENT()              user_callback_suspend_action()
// extern void user_callback_suspend_action(void);
// #define  UDC_RESUME_EVENT()               user_callback_resume_action()
// extern void user_callback_resume_action(void);
// #define  UDC_REMOTEWAKEUP_ENABLE()        user_callback_remotewakeup_enable()
// extern void user_callback_remotewakeup_enable(void);
// #define  UDC_REMOTEWAKEUP_DISABLE()       user_callback_remotewakeup_disable()
// extern void user_callback_remotewakeup_disable(void);
// #define  UDC_GET_EXTRA_STRING()

#define  USB_DEVICE_EP_CTRL_SIZE       64

#define  USB_DEVICE_NB_INTERFACE       1 // 1 or more

#define  USB_DEVICE_MAX_EP             1 // 0 to max endpoint requested by interfaces




#define  UDI_CDC_PORT_NB 1

#define  UDI_CDC_ENABLE_EXT(port)             true
#define  UDI_CDC_DISABLE_EXT(port)
#define  UDI_CDC_RX_NOTIFY(port)
#define  UDI_CDC_TX_EMPTY_NOTIFY(port)
#define  UDI_CDC_SET_CODING_EXT(port,cfg)
#define  UDI_CDC_SET_DTR_EXT(port,set)
#define  UDI_CDC_SET_RTS_EXT(port,set)
/*
 * #define UDI_CDC_ENABLE_EXT(port) my_callback_cdc_enable()
 * extern bool my_callback_cdc_enable(void);
 * #define UDI_CDC_DISABLE_EXT(port) my_callback_cdc_disable()
 * extern void my_callback_cdc_disable(void);
 * #define  UDI_CDC_RX_NOTIFY(port) my_callback_rx_notify(port)
 * extern void my_callback_rx_notify(uint8_t port);
 * #define  UDI_CDC_TX_EMPTY_NOTIFY(port) my_callback_tx_empty_notify(port)
 * extern void my_callback_tx_empty_notify(uint8_t port);
 * #define  UDI_CDC_SET_CODING_EXT(port,cfg) my_callback_config(port,cfg)
 * extern void my_callback_config(uint8_t port, usb_cdc_line_coding_t * cfg);
 * #define  UDI_CDC_SET_DTR_EXT(port,set) my_callback_cdc_set_dtr(port,set)
 * extern void my_callback_cdc_set_dtr(uint8_t port, bool b_enable);
 * #define  UDI_CDC_SET_RTS_EXT(port,set) my_callback_cdc_set_rts(port,set)
 * extern void my_callback_cdc_set_rts(uint8_t port, bool b_enable);
 */

#define  UDI_CDC_LOW_RATE

#define  UDI_CDC_DEFAULT_RATE             115200
#define  UDI_CDC_DEFAULT_STOPBITS         CDC_STOP_BITS_1
#define  UDI_CDC_DEFAULT_PARITY           CDC_PAR_NONE
#define  UDI_CDC_DEFAULT_DATABITS         8

#define  UDI_CDC_DATA_EP_IN_0         (1 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_0        (2 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_0            (3 | USB_EP_DIR_IN)  // Notify endpoint
#define  UDI_CDC_DATA_EP_IN_2         (4 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_2        (5 | USB_EP_DIR_OUT) // RX
#define  UDI_CDC_COMM_EP_2            (6 | USB_EP_DIR_IN)  // Notify endpoint
#define  UDI_CDC_DATA_EP_IN_3         (7 | USB_EP_DIR_IN)  // TX
#define  UDI_CDC_DATA_EP_OUT_3        (8 | USB_EP_DIR_OUT) // RX
```

```
#define  UDI_CDC_COMM_EP_3            (9 | USB_EP_DIR_IN)  // Notify endpoint

#define  UDI_CDC_COMM_IFACE_NUMBER_0  0
#define  UDI_CDC_DATA_IFACE_NUMBER_0  1
#define  UDI_CDC_COMM_IFACE_NUMBER_2  2
#define  UDI_CDC_DATA_IFACE_NUMBER_2  3
#define  UDI_CDC_COMM_IFACE_NUMBER_3  4
#define  UDI_CDC_DATA_IFACE_NUMBER_3  5




#define UDI_MSC_GLOBAL_VENDOR_ID           \
   'A', 'T', 'M', 'E', 'L', ' ', ' ', ' '
#define UDI_MSC_GLOBAL_PRODUCT_VERSION           \
   '1', '.', '0', '0'

#define  UDI_MSC_ENABLE_EXT()         true
#define  UDI_MSC_DISABLE_EXT()
#define  UDI_MSC_NOTIFY_TRANS_EXT()
/*
 * #define UDI_MSC_ENABLE_EXT() my_callback_msc_enable()
 * extern bool my_callback_msc_enable(void);
 * #define UDI_MSC_DISABLE_EXT() my_callback_msc_disable()
 * extern void my_callback_msc_disable(void);
 * #define  UDI_MSC_NOTIFY_TRANS_EXT()    msc_notify_trans()
 * extern void msc_notify_trans(void) {
 */

#define  UDI_MSC_EP_IN                (1 | USB_EP_DIR_IN)
#define  UDI_MSC_EP_OUT               (2 | USB_EP_DIR_OUT)

#define  UDI_MSC_IFACE_NUMBER         0




#define  UDI_HID_MOUSE_ENABLE_EXT()       true
#define  UDI_HID_MOUSE_DISABLE_EXT()
// #define UDI_HID_MOUSE_ENABLE_EXT() my_callback_mouse_enable()
// extern bool my_callback_mouse_enable(void);
// #define UDI_HID_MOUSE_DISABLE_EXT() my_callback_mouse_disable()
// extern void my_callback_mouse_disable(void);

#define  UDI_HID_MOUSE_EP_IN          (1 | USB_EP_DIR_IN)

#define  UDI_HID_MOUSE_IFACE_NUMBER    0




#define  UDI_HID_KBD_ENABLE_EXT()       true
#define  UDI_HID_KBD_DISABLE_EXT()
// #define UDI_HID_KBD_ENABLE_EXT() my_callback_keyboard_enable()
// extern bool my_callback_keyboard_enable(void);
// #define UDI_HID_KBD_DISABLE_EXT() my_callback_keyboard_disable()
// extern void my_callback_keyboard_disable(void);
#define  UDI_HID_KBD_CHANGE_LED(value)
// #define  UDI_HID_KBD_CHANGE_LED(value) my_callback_keyboard_led(value)
// extern void my_callback_keyboard_led(uint8_t value)
```

```
#define  UDI_HID_KBD_EP_IN           (1 | USB_EP_DIR_IN)

#define  UDI_HID_KBD_IFACE_NUMBER    0



#define  UDI_HID_GENERIC_ENABLE_EXT()       true
#define  UDI_HID_GENERIC_DISABLE_EXT()
#define  UDI_HID_GENERIC_REPORT_OUT(ptr)
#define  UDI_HID_GENERIC_SET_FEATURE(f)
/*
 * #define UDI_HID_GENERIC_ENABLE_EXT() my_callback_generic_enable()
 * extern bool my_callback_generic_enable(void);
 * #define UDI_HID_GENERIC_DISABLE_EXT() my_callback_generic_disable()
 * extern void my_callback_generic_disable(void);
 * #define  UDI_HID_GENERIC_REPORT_OUT(ptr) my_callback_generic_report_out(ptr)
 * extern void my_callback_generic_report_out(uint8_t *report);
 * #define  UDI_HID_GENERIC_SET_FEATURE(f) my_callback_generic_set_feature(f)
 * extern void my_callback_generic_set_feature(uint8_t *report_feature);
 */
#define  UDI_HID_REPORT_IN_SIZE             64
#define  UDI_HID_REPORT_OUT_SIZE            64
#define  UDI_HID_REPORT_FEATURE_SIZE        4
#define  UDI_HID_GENERIC_EP_SIZE            64

#define  UDI_HID_GENERIC_EP_OUT   (2 | USB_EP_DIR_OUT)
#define  UDI_HID_GENERIC_EP_IN    (1 | USB_EP_DIR_IN)

#define  UDI_HID_GENERIC_IFACE_NUMBER    0



#define  UDI_PHDC_ENABLE_EXT()          true
#define  UDI_PHDC_DISABLE_EXT()

#define  UDI_PHDC_DATAMSG_FORMAT       USB_PHDC_DATAMSG_FORMAT_11073_20601
#define  UDI_PHDC_SPECIALIZATION       {0x2345} // Define in 11073_20601

#define  UDI_PHDC_QOS_OUT              \
        (USB_PHDC_QOS_MEDIUM_BETTER|USB_PHDC_QOS_HIGH_BEST)
#define  UDI_PHDC_QOS_IN               \
        (USB_PHDC_QOS_LOW_GOOD|USB_PHDC_QOS_MEDIUM_BETTER|USB_PHDC_QOS_MEDIUM_BEST)

#define  UDI_PHDC_METADATA_DESC_BULK_IN   {0x01,0x02,0x03}
#define  UDI_PHDC_METADATA_DESC_BULK_OUT  {0x01,0x02,0x03}
#define  UDI_PHDC_METADATA_DESC_INT_IN    {0x01,0x02,0x03}

#define  UDI_PHDC_EP_BULK_OUT          (1 | USB_EP_DIR_OUT)
#define  UDI_PHDC_EP_BULK_IN           (2 | USB_EP_DIR_IN)
#if ((UDI_PHDC_QOS_IN&USB_PHDC_QOS_LOW_GOOD)==USB_PHDC_QOS_LOW_GOOD)
// Only if UDI_PHDC_QOS_IN include USB_PHDC_QOS_LOW_GOOD
#  define  UDI_PHDC_EP_INTERRUPT_IN      (3 | USB_EP_DIR_IN)
#endif

#define  UDI_PHDC_EP_SIZE_BULK_OUT     32
#define  UDI_PHDC_EP_SIZE_BULK_IN      32
#define  UDI_PHDC_EP_SIZE_INT_IN       8
```

```
#define  UDI_PHDC_IFACE_NUMBER          0



#define  UDI_VENDOR_ENABLE_EXT()          true
#define  UDI_VENDOR_DISABLE_EXT()
#define  UDI_VENDOR_SETUP_OUT_RECEIVED()  false
#define  UDI_VENDOR_SETUP_IN_RECEIVED()   false
/*
 * #define UDI_VENDOR_ENABLE_EXT() my_callback_vendor_enable()
 * extern bool my_callback_vendor_enable(void);
 * #define UDI_VENDOR_DISABLE_EXT() my_callback_vendor_disable()
 * extern void my_callback_vendor_disable(void);
 *
 * #define  UDI_VENDOR_SETUP_OUT_RECEIVED()  my_vendor_setup_out_received()
 * extern bool my_vendor_setup_out_received(void);
 * #define  UDI_VENDOR_SETUP_IN_RECEIVED()   my_vendor_setup_in_received()
 * extern bool my_vendor_setup_in_received(void);
 */

#define  UDI_VENDOR_EPS_SIZE_INT_FS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_FS   64
#define  UDI_VENDOR_EPS_SIZE_ISO_FS   256

#define  UDI_VENDOR_EPS_SIZE_INT_HS    64
#define  UDI_VENDOR_EPS_SIZE_BULK_HS  512
#define  UDI_VENDOR_EPS_SIZE_ISO_HS    64

#define  UDI_VENDOR_EP_INTERRUPT_IN  (1 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_INTERRUPT_OUT (2 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_BULK_IN       (3 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_BULK_OUT      (4 | USB_EP_DIR_OUT)
#define  UDI_VENDOR_EP_ISO_IN        (5 | USB_EP_DIR_IN)
#define  UDI_VENDOR_EP_ISO_OUT       (6 | USB_EP_DIR_OUT)


#define  UDI_VENDOR_IFACE_NUMBER     0



//... Eventually add other Interface Configuration



#define UDI_COMPOSITE_DESC_T

#define UDI_COMPOSITE_DESC_FS

#define UDI_COMPOSITE_DESC_HS

#define UDI_COMPOSITE_API

/* Example for device with cdc, msc and hid mouse interface
#define UDI_COMPOSITE_DESC_T \
    usb_iad_desc_t udi_cdc_iad; \
    udi_cdc_comm_desc_t udi_cdc_comm; \
    udi_cdc_data_desc_t udi_cdc_data; \
    udi_msc_desc_t udi_msc; \
    udi_hid_mouse_desc_t udi_hid_mouse

#define UDI_COMPOSITE_DESC_FS \
```

```
    .udi_cdc_iad               = UDI_CDC_IAD_DESC_0, \
    .udi_cdc_comm              = UDI_CDC_COMM_DESC_0, \
    .udi_cdc_data              = UDI_CDC_DATA_DESC_0_FS, \
    .udi_msc                   = UDI_MSC_DESC_FS, \
    .udi_hid_mouse             = UDI_HID_MOUSE_DESC

#define UDI_COMPOSITE_DESC_HS \
    .udi_cdc_iad               = UDI_CDC_IAD_DESC_0, \
    .udi_cdc_comm              = UDI_CDC_COMM_DESC_0, \
    .udi_cdc_data              = UDI_CDC_DATA_DESC_0_HS, \
    .udi_msc                   = UDI_MSC_DESC_HS, \
    .udi_hid_mouse             = UDI_HID_MOUSE_DESC

#define UDI_COMPOSITE_API  \
    &udi_api_cdc_comm,       \
    &udi_api_cdc_data,       \
    &udi_api_msc,            \
    &udi_api_hid_mouse
*/




/* Example of include for interface
#include "udi_msc.h"
#include "udi_hid_kbd.h"
#include "udi_hid_mouse.h"
#include "udi_cdc.h"
#include "udi_phdc.h"
#include "udi_vendor.h"
*/
/* Declaration of callbacks used by USB
#include "callback_def.h"
*/

#endif // _CONF_USB_H_
```

7.3.2 conf_clock.h

7.3.2.1 SAM3X and SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock (MCK) Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_BYPASS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_4M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_8M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_12M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_BYPASS
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLLACK
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_UPLLCK

// ===== System Clock (MCK) Prescaler Options   (Fmck = Fsys / (SYSCLK_PRES))
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_1
#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_2
```

```
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_4
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_8
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_16
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_32
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_64
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_3

// ===== PLL0 (A) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
// Use mul and div effective values here.
#define CONFIG_PLL0_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL0_MUL             14
#define CONFIG_PLL0_DIV             1

// ===== UPLL (UTMI) Hardware fixed at 480MHz.

// ===== USB Clock Source Options   (Fusb = FpllX / USB_div)
// Use div effective value here.
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_UPLL
#define CONFIG_USBCLK_DIV           1

// ===== Target frequency (System clock)
// - XTAL frequency: 12MHz
// - System clock source: PLLA
// - System clock prescaler: 2 (divided by 2)
// - PLLA source: XTAL
// - PLLA output: XTAL * 14 / 1
// - System clock is: 12 * 14 / 1 /2 = 84MHz
// ===== Target frequency (USB Clock)
// - USB clock source: UPLL
// - USB clock divider: 1 (not divided)
// - UPLL frequency: 480MHz
// - USB clock: 480 / 1 = 480MHz


#endif /* CONF_CLOCK_H_INCLUDED */
```

7.3.2.2 SAM4L Device (USBC)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

//#define CONFIG_SYSCLK_INIT_CPUMASK  (1 << SYSCLK_OCD)
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_IISC)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_USBC_REGS)
//#define CONFIG_SYSCLK_INIT_PBCMASK  (1 << SYSCLK_CHIPID)
//#define CONFIG_SYSCLK_INIT_PBDMASK  (1 << SYSCLK_AST)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_PDCA_HSB)

//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_DFLL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RC80M
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCFAST
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RC1M

/* RCFAST frequency selection: 0 for 4MHz, 1 for 8MHz and 2 for 12MHz */
//#define CONFIG_RCFAST_FRANGE    0
//#define CONFIG_RCFAST_FRANGE    1
```

```
//#define CONFIG_RCFAST_FRANGE    2

/* Fbus = Fsys / (2 ^ BUS_div) */
#define CONFIG_SYSCLK_CPU_DIV         0
#define CONFIG_SYSCLK_PBA_DIV         0
#define CONFIG_SYSCLK_PBB_DIV         0
#define CONFIG_SYSCLK_PBC_DIV         0
#define CONFIG_SYSCLK_PBD_DIV         0

// ===== Disable all non-essential peripheral clocks
//#define CONFIG_SYSCLK_INIT_CPUMASK  0
//#define CONFIG_SYSCLK_INIT_PBAMASK  SYSCLK_USART1
//#define CONFIG_SYSCLK_INIT_PBBMASK  0
//#define CONFIG_SYSCLK_INIT_PBCMASK  0
//#define CONFIG_SYSCLK_INIT_PBDMASK  0
//#define CONFIG_SYSCLK_INIT_HSBMASK  0

// ===== PLL Options
#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_GCLK9

/* Fpll0 = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL0_MUL             (48000000UL / BOARD_OSC0_HZ)
#define CONFIG_PLL0_DIV             1
//#define CONFIG_PLL0_MUL             (192000000 / FOSC0) /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL0_DIV             4 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ==== DFLL Options
//#define CONFIG_DFLL0_SOURCE         GENCLK_SRC_OSC0
//#define CONFIG_DFLL0_SOURCE         GENCLK_SRC_RCSYS
//#define CONFIG_DFLL0_SOURCE         GENCLK_SRC_OSC32K
//#define CONFIG_DFLL0_SOURCE         GENCLK_SRC_RC120M
//#define CONFIG_DFLL0_SOURCE         GENCLK_SRC_RC32K

/* Fdfll = (Fclk * DFLL_mul) / DFLL_div */
//#define CONFIG_DFLL0_FREQ           48000000UL
//#define CONFIG_DFLL0_MUL            ((4 * CONFIG_DFLL0_FREQ) / BOARD_OSC32_HZ)
//#define CONFIG_DFLL0_DIV            4
//#define CONFIG_DFLL0_MUL            (CONFIG_DFLL0_FREQ / BOARD_OSC32_HZ)
//#define CONFIG_DFLL0_DIV            1

// ===== USB Clock Source Options
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_DFLL

/* Fusb = Fsys / USB_div */
#define CONFIG_USBCLK_DIV           1

// ===== GCLK9 option
//#define CONFIG_GCLK9_SOURCE           GENCLK_SRC_GCLKIN0
//#define CONFIG_GCLK9_DIV              1

#endif /* CONF_CLOCK_H_INCLUDED */
```

7.3.3 conf_clocks.h

7.3.3.1 SAMD21 Device (USB)

```
#include <clock.h>
```

```
#ifndef CONF_CLOCKS_H_INCLUDED
#  define CONF_CLOCKS_H_INCLUDED

/* System clock bus configuration */
#  define CONF_CLOCK_CPU_CLOCK_FAILURE_DETECT     false
#  define CONF_CLOCK_FLASH_WAIT_STATES            2
#  define CONF_CLOCK_CPU_DIVIDER                  SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBA_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBB_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1

/* SYSTEM_CLOCK_SOURCE_OSC8M configuration - Internal 8MHz oscillator */
#  define CONF_CLOCK_OSC8M_PRESCALER              SYSTEM_OSC8M_DIV_1
#  define CONF_CLOCK_OSC8M_ON_DEMAND              true
#  define CONF_CLOCK_OSC8M_RUN_IN_STANDBY         false

/* SYSTEM_CLOCK_SOURCE_XOSC configuration - External clock/oscillator */
#  define CONF_CLOCK_XOSC_ENABLE                  false
#  define CONF_CLOCK_XOSC_EXTERNAL_CRYSTAL        SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC_EXTERNAL_FREQUENCY      12000000UL
#  define CONF_CLOCK_XOSC_STARTUP_TIME            SYSTEM_XOSC_STARTUP_32768
#  define CONF_CLOCK_XOSC_AUTO_GAIN_CONTROL       true
#  define CONF_CLOCK_XOSC_ON_DEMAND               true
#  define CONF_CLOCK_XOSC_RUN_IN_STANDBY          false

/* SYSTEM_CLOCK_SOURCE_XOSC32K configuration - External 32KHz crystal/clock oscillator */
#  define CONF_CLOCK_XOSC32K_ENABLE               false
#  define CONF_CLOCK_XOSC32K_EXTERNAL_CRYSTAL     SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC32K_STARTUP_TIME         SYSTEM_XOSC32K_STARTUP_65536
#  define CONF_CLOCK_XOSC32K_AUTO_AMPLITUDE_CONTROL  false
#  define CONF_CLOCK_XOSC32K_ENABLE_1KHZ_OUPUT    false
#  define CONF_CLOCK_XOSC32K_ENABLE_32KHZ_OUTPUT  true
#  define CONF_CLOCK_XOSC32K_ON_DEMAND            true
#  define CONF_CLOCK_XOSC32K_RUN_IN_STANDBY       false

/* SYSTEM_CLOCK_SOURCE_OSC32K configuration - Internal 32KHz oscillator */
#  define CONF_CLOCK_OSC32K_ENABLE                false
#  define CONF_CLOCK_OSC32K_STARTUP_TIME          SYSTEM_OSC32K_STARTUP_130
#  define CONF_CLOCK_OSC32K_ENABLE_1KHZ_OUTPUT    true
#  define CONF_CLOCK_OSC32K_ENABLE_32KHZ_OUTPUT   true
#  define CONF_CLOCK_OSC32K_ON_DEMAND             true
#  define CONF_CLOCK_OSC32K_RUN_IN_STANDBY        false

/* SYSTEM_CLOCK_SOURCE_DFLL configuration - Digital Frequency Locked Loop */
#  define CONF_CLOCK_DFLL_ENABLE                  true
#  define CONF_CLOCK_DFLL_LOOP_MODE               SYSTEM_CLOCK_DFLL_LOOP_MODE_USB_RECOVERY
#  define CONF_CLOCK_DFLL_ON_DEMAND               true

/* DFLL open loop mode configuration */
#  define CONF_CLOCK_DFLL_COARSE_VALUE            (0x1f / 4)
#  define CONF_CLOCK_DFLL_FINE_VALUE              (0xff / 4)

/* DFLL closed loop mode configuration */
#  define CONF_CLOCK_DFLL_SOURCE_GCLK_GENERATOR   GCLK_GENERATOR_1
#  define CONF_CLOCK_DFLL_MULTIPLY_FACTOR         (48000000 / 32768)
#  define CONF_CLOCK_DFLL_QUICK_LOCK              true
#  define CONF_CLOCK_DFLL_TRACK_AFTER_FINE_LOCK   true
#  define CONF_CLOCK_DFLL_KEEP_LOCK_ON_WAKEUP     true
#  define CONF_CLOCK_DFLL_ENABLE_CHILL_CYCLE      true
#  define CONF_CLOCK_DFLL_MAX_COARSE_STEP_SIZE    (0x1f / 4)
#  define CONF_CLOCK_DFLL_MAX_FINE_STEP_SIZE      (0xff / 4)
```

```
/* SYSTEM_CLOCK_SOURCE_DPLL configuration - Digital Phase-Locked Loop */
#  define CONF_CLOCK_DPLL_ENABLE                    false
#  define CONF_CLOCK_DPLL_ON_DEMAND                 true
#  define CONF_CLOCK_DPLL_RUN_IN_STANDBY            false
#  define CONF_CLOCK_DPLL_LOCK_BYPASS               false
#  define CONF_CLOCK_DPLL_WAKE_UP_FAST              false
#  define CONF_CLOCK_DPLL_LOW_POWER_ENABLE          false

#  define CONF_CLOCK_DPLL_LOCK_TIME                 SYSTEM_CLOCK_SOURCE_DPLL_LOCK_TIME_NO_TIMEOUT
#  define CONF_CLOCK_DPLL_REFERENCE_CLOCK           SYSTEM_CLOCK_SOURCE_DPLL_REFERENCE_CLOCK_REF0
#  define CONF_CLOCK_DPLL_FILTER                    SYSTEM_CLOCK_SOURCE_DPLL_FILTER_DEFAULT

#  define CONF_CLOCK_DPLL_REFERENCE_FREQUENCY       32768
#  define CONF_CLOCK_DPLL_REFEREMCE_DIVIDER         1
#  define CONF_CLOCK_DPLL_OUTPUT_FREQUENCY          48000000

/* Set this to true to configure the GCLK when running clocks_init. If set to
 * false, none of the GCLK generators will be configured in clocks_init(). */
#  define CONF_CLOCK_CONFIGURE_GCLK                 true

/* Configure GCLK generator 0 (Main Clock) */
#  define CONF_CLOCK_GCLK_0_ENABLE                  true
#  define CONF_CLOCK_GCLK_0_RUN_IN_STANDBY          true
#  define CONF_CLOCK_GCLK_0_CLOCK_SOURCE            SYSTEM_CLOCK_SOURCE_DFLL
#  define CONF_CLOCK_GCLK_0_PRESCALER               1
#  define CONF_CLOCK_GCLK_0_OUTPUT_ENABLE           false

/* Configure GCLK generator 1 */
#  define CONF_CLOCK_GCLK_1_ENABLE                  false
#  define CONF_CLOCK_GCLK_1_RUN_IN_STANDBY          false
#  define CONF_CLOCK_GCLK_1_CLOCK_SOURCE            SYSTEM_CLOCK_SOURCE_XOSC32K
#  define CONF_CLOCK_GCLK_1_PRESCALER               1
#  define CONF_CLOCK_GCLK_1_OUTPUT_ENABLE           false

/* Configure GCLK generator 2 (RTC) */
#  define CONF_CLOCK_GCLK_2_ENABLE                  false
#  define CONF_CLOCK_GCLK_2_RUN_IN_STANDBY          false
#  define CONF_CLOCK_GCLK_2_CLOCK_SOURCE            SYSTEM_CLOCK_SOURCE_OSC32K
#  define CONF_CLOCK_GCLK_2_PRESCALER               32
#  define CONF_CLOCK_GCLK_2_OUTPUT_ENABLE           false

/* Configure GCLK generator 3 */
#  define CONF_CLOCK_GCLK_3_ENABLE                  false
#  define CONF_CLOCK_GCLK_3_RUN_IN_STANDBY          false
#  define CONF_CLOCK_GCLK_3_CLOCK_SOURCE            SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_3_PRESCALER               1
#  define CONF_CLOCK_GCLK_3_OUTPUT_ENABLE           false

/* Configure GCLK generator 4 */
#  define CONF_CLOCK_GCLK_4_ENABLE                  false
#  define CONF_CLOCK_GCLK_4_RUN_IN_STANDBY          false
#  define CONF_CLOCK_GCLK_4_CLOCK_SOURCE            SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_4_PRESCALER               1
#  define CONF_CLOCK_GCLK_4_OUTPUT_ENABLE           false

/* Configure GCLK generator 5 */
#  define CONF_CLOCK_GCLK_5_ENABLE                  false
#  define CONF_CLOCK_GCLK_5_RUN_IN_STANDBY          false
#  define CONF_CLOCK_GCLK_5_CLOCK_SOURCE            SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_5_PRESCALER               1
#  define CONF_CLOCK_GCLK_5_OUTPUT_ENABLE           false
```

```
/* Configure GCLK generator 6 */
#  define CONF_CLOCK_GCLK_6_ENABLE                 false
#  define CONF_CLOCK_GCLK_6_RUN_IN_STANDBY         false
#  define CONF_CLOCK_GCLK_6_CLOCK_SOURCE           SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_6_PRESCALER              1
#  define CONF_CLOCK_GCLK_6_OUTPUT_ENABLE          false

/* Configure GCLK generator 7 */
#  define CONF_CLOCK_GCLK_7_ENABLE                 false
#  define CONF_CLOCK_GCLK_7_RUN_IN_STANDBY         false
#  define CONF_CLOCK_GCLK_7_CLOCK_SOURCE           SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_7_PRESCALER              1
#  define CONF_CLOCK_GCLK_7_OUTPUT_ENABLE          false

#endif /* CONF_CLOCKS_H_INCLUDED */
```

7.3.4 conf_board.h

7.3.4.1 SAM3X and SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// USB pins are used
#define CONF_BOARD_USB_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

7.3.4.2 SAM4L Device (USBC)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Auto-initialize USART GPIOs when board_init() is called
//#define CONF_BOARD_COM_PORT

// Enable USB interface (USB)
#define CONF_BOARD_USB_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

7.3.4.3 SAMD21 Device (USB)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

/* Enable USB VBUS detect */
#define CONF_BOARD_USB_VBUS_DETECT

#endif /* CONF_BOARD_H_INCLUDED */
```

8. USB Host Controller (UHC)

The UHC provides a high-level abstraction of the usb host. You can use these functions to control the main host state (start/suspend/resume/...).

All USB Host Interface (UHI) in USB Host Stack is based on UHC to support USB enumeration.

For more details for Atmel Software Framework (ASF) USB Host Stack, refer to following application note:

- AVR4950: ASF - USB Host Stack[1]

This documentation describes common USB Host usage based on UHC, as follow:

- API Overview
- USB Host Basic Setup
- USB Host Advanced Use Cases

8.1 API Overview

8.1.1 Structure Definitions

8.1.1.1 Struct uhc_device_t

Note The fields of this structure should not be altered by the user application; they are reserved only for module-internal use.

Table 8-1. Members

Type	Name	Description
uint8_t	address	USB address
usb_conf_desc_t *	conf_desc	USB current configuration descriptor
usb_dev_desc_t	dev_desc	USB device descriptor
uhd_speed_t	speed	USB speed

8.1.2 Function Definitions

8.1.2.1 Functions to Control the USB Host Stack

Function uhc_start()

Starts the host mode.

```
void uhc_start(void)
```

Function uhc_stop()

Stops the host mode.

```
void uhc_stop(
```

[1] http://www.atmel.com/dyn/resources/prod_documents/doc8486.pdf

```
  bool b_id_stop)
```

Table 8-2. Parameters

Data direction	Parameter name	Description
[in]	b_id_stop	Stop USB ID pin management, if true.

Function uhc_suspend()

Suspends a USB line.

```
void uhc_suspend(
  bool b_remotewakeup)
```

Table 8-3. Parameters

Data direction	Parameter name	Description
[in]	b_remotewakeup	Authorize the remote wakeup features, if true.

Function uhc_is_suspend()

Test if the suspend state is enabled on the USB line.

```
bool uhc_is_suspend(void)
```

Returns USB line in SUSPEND state or device not connected, if true.

Function uhc_resume()

Resumes the USB line.

```
void uhc_resume(void)
```

Function uhc_suspend_lpm()

Suspends a USB line through LPM feature(SAM D21).

```
bool uhc_suspend_lpm(
  bool b_remotewakeup,
  uint8_t besl)
```

Table 8-4. Parameters

Data direction	Parameter name	Description
[in]	b_remotewakeup	Authorize the remote wakeup features, if true.

Data direction	Parameter name	Description
[in]	besl	Best effort service latency value.

Returns False if the LPM is not supported by USB Device.

8.1.2.2 User Functions to Manage a Device

Function uhc_get_device_number()

Returns the number of connected devices.

```
uint8_t uhc_get_device_number(void)
```

Returns Number of device connected on USB tree.

Function uhc_dev_get_string_manufacturer()

Gets the USB string manufacturer from a USB device.

```
char * uhc_dev_get_string_manufacturer(
  uhc_device_t * dev)
```

This function waits the end of setup requests and the timing can be long (3ms to 15s). Thus, do not call it in an interrupt routine. This function allocates a buffer which must be free by user application.

Table 8-5. Parameters

Data direction	Parameter name	Description
[in]	dev	Device to request.

Returns Pointer on unicode string, or NULL if function fails.

Function uhc_dev_get_string_product()

Gets the USB string product from a USB device.

```
char * uhc_dev_get_string_product(
  uhc_device_t * dev)
```

This function waits the end of setup requests and the timing can be long (3ms to 15s). Thus, do not call it in an interrupt routine. This function allocates a buffer which must be free by user application.

Table 8-6. Parameters

Data direction	Parameter name	Description
[in]	dev	Device to request.

Returns Pointer on unicode string, or NULL if function fails.

Function uhc_dev_get_string_serial()

Gets the USB string serial from a USB device.

```
char * uhc_dev_get_string_serial(
  uhc_device_t * dev)
```

This function waits the end of setup requests and the timing can be long (3ms to 15s). Thus, do not call it in an interrupt routine. This function allocates a buffer which must be free by user application.

Table 8-7. Parameters

Data direction	Parameter name	Description
[in]	dev	Device to request.

Returns Pointer on unicode string, or NULL if function fails.

Function uhc_dev_get_string()

Gets a USB string from a USB device.

```
char * uhc_dev_get_string(
  uhc_device_t * dev,
  uint8_t str_id)
```

This function waits the end of setup requests and the timing can be long (3ms to 15s). Thus, do not call it in an interrupt routine. This function allocates a buffer which must be free by user application.

Table 8-8. Parameters

Data direction	Parameter name	Description
[in]	dev	Device to request.
[in]	str_id	String ID requested.

Returns Pointer on unicode string, or NULL if function fails.

Function uhc_dev_get_power()

Gets the maximum consumption of a device (mA).

```
uint16_t uhc_dev_get_power(
  uhc_device_t * dev)
```

Table 8-9. Parameters

Data direction	Parameter name	Description
[in]	dev	Device to request.

Returns Maximum consumption of the device (mA).

Function uhc_dev_get_speed()

Returns the current device speed.

```
uhd_speed_t uhc_dev_get_speed(
  uhc_device_t * dev)
```

Table 8-10. Parameters

Data direction	Parameter name	Description
[in]	dev	Device to request.

Returns Device speed.

Function uhc_dev_is_high_speed_support()

Tests if the device supports the USB high speed. This function can wait the end of a setup request and the timing can be long (1ms to 5s). Thus, do not call it in an interrupt routine.

```
bool uhc_dev_is_high_speed_support(
  uhc_device_t * dev)
```

Table 8-11. Parameters

Data direction	Parameter name	Description
[in]	dev	Device to request.

Returns True, if high speed is supported.

8.1.3 Enumeration Definitions

8.1.3.1 Enum uhc_enum_status_t

Table 8-12. Members

Enum value	Description
UHC_ENUM_SUCCESS	Device is enumerated. The supported USB device interfaces has been enabled.
UHC_ENUM_UNSUPPORTED	None of the interfaces are supported by the UHIs.
UHC_ENUM_OVERCURRENT	Device power is not supported.
UHC_ENUM_FAIL	A problem occurred during USB enumeration.
UHC_ENUM_HARDWARE_LIMIT	USB hardware can not support it. Not enough free pipes.

Enum value	Description
UHC_ENUM_SOFTWARE_LIMIT	USB software can not support it. Implementation limit.
UHC_ENUM_MEMORY_LIMIT	USB software can not support it. Not enough memory.
UHC_ENUM_DISCONNECT	The device has been disconnected during USB enumeration.

8.2 USB Host Basic Setup

8.2.1 USB Host User Configuration

The following USB host configuration must be included in the conf_usb_host.h file of the application:

1. USB_HOST_UHI (List of UHI APIs).

Define the list of UHI supported by USB host. (E.g.: UHI_MSC, UHI_HID_MOUSE).

2. USB_HOST_POWER_MAX (mA).

Maximum current allowed on Vbus.

3. USB_HOST_HS_SUPPORT (Only defined).

Authorize the USB host to run in High Speed.

4. USB_HOST_HUB_SUPPORT (Only defined).

Authorize the USB HUB support.

8.2.2 USB Host User Callback

The following optional USB host callback can be defined in the conf_usb_host.h file of the application:

1. void UHC_MODE_CHANGE(bool b_host_mode).

To notify that the USB mode are switched automatically. This is possible only when ID pin is available.

2. void UHC_VBUS_CHANGE(bool b_present).

To notify that the Vbus level has changed (Available only in USB hardware with Vbus monitoring).

3. void UHC_VBUS_ERROR(void).

To notify that a Vbus error has occurred (Available only in USB hardware with Vbus monitoring).

4. void UHC_CONNECTION_EVENT(uhc_device_t* dev, bool b_present).

To notify that a device has been connected or disconnected.

5. void UHC_WAKEUP_EVENT(void).

Called when a USB device or the host have wake up the USB line.

6. void UHC_SOF_EVENT(void).

Called for each received SOF each 1ms. Available in High and Full speed mode.

7. uint8_t UHC_DEVICE_CONF(uhc_device_t* dev).

Called when a USB device configuration must be chosen. Thus, the application can choose either a configuration number for this device or a configuration number 0 to reject it. If callback not defined the configuration 1 is chosen.

8. void UHC_ENUM_EVENT(uhc_device_t* dev, uint8_t b_status).

Called when a USB device enumeration is completed or failed.

8.2.3 USB Host Setup Steps

8.2.3.1 USB Host Controller (UHC) - Prerequisites

Common prerequisites for all USB hosts.

This module is based on USB host stack full interrupt driven and supporting sleepmgr. For AVR® and Atmel® | SMART SAM3/4 devices the clock services is supported. For SAMD21 devices the clock driver is supported.

The following procedure must be executed to setup the project correctly:

- Specify the clock configuration:
  - UC3 and SAM3/4 devices without USB high speed support need 48MHz clock input.
    You must use a PLL and an external OSC.
  - UC3 and SAM3/4 devices with USB high speed support need 12MHz clock input.
    You must use an external OSC.
  - UC3 devices with USBC hardware need CPU frequency higher than 25MHz.
  - SAMD21 devices without USB high speed support need 48MHz clock input.
    You must use a DFLL and an external OSC.
- In conf_board.h, the define CONF_BOARD_USB_PORT must be added to enable USB lines. (Not mandatory for all boards)
- Enable interrupts
- Initialize the clock service

The usage of sleepmgr service is optional, but recommended to reduce power consumption:

- Initialize the sleep manager service
- Activate sleep mode when the application is in IDLE state

conf_clock.h Examples.

For AVR and SAM3/4 devices, add to the initialization code:

```
sysclk_init();
irq_initialize_vectors();
cpu_irq_enable();
board_init();
sleepmgr_init(); // Optional
```

For SAMD21 devices, add to the initialization code:

```
system_init();
irq_initialize_vectors();
cpu_irq_enable();
sleepmgr_init(); // Optional
```

Add to the main IDLE loop:

```
sleepmgr_enter_sleep(); // Optional
```

8.2.3.2 USB Host Controller (UHC) - Example Code

Common example code for all USB hosts.

Content of conf_usb_host.h:

```
#define USB_HOST_POWER_MAX  500
```

Add to application C-file:

```
void usb_init(void)
{
    uhc_start();
}
```

8.2.3.3 USB Device Controller (UHC) - Workflow

Common workflow for all USB devices.

1. Ensure that conf_usb_host.h is available and contains the following configuration which is the main USB device configuration:

```
// Maximum current allowed on Vbus (mA) which depends of 5V generator
#define USB_HOST_POWER_MAX  500 // (500mA)
```

2. Call the USB host stack start function to enable USB Host stack:

```
uhc_start();
```

8.2.4 conf_clock.h Examples

Content of conf_clock.h for AT32UC3A0, AT32UC3A1, and AT32UC3B devices (USBB):

```
// Configuration based on 12MHz external OSC:
#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
#define CONFIG_PLL1_MUL             8
#define CONFIG_PLL1_DIV             2
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV           1 // Fusb = Fsys/(2 ^ USB_div)
```

Content of conf_clock.h for AT32UC3A3 and AT32UC3A4 devices (USBB with high speed support):

```
// Configuration based on 12MHz external OSC:
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
#define CONFIG_USBCLK_DIV           1 // Fusb = Fsys/(2 ^ USB_div)
```

Content of conf_clock.h for AT32UC3C device (USBC):

```
// Configuration based on 12MHz external OSC:
#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
#define CONFIG_PLL1_MUL             8
#define CONFIG_PLL1_DIV             2
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV           1 // Fusb = Fsys/(2 ^ USB_div)
// CPU clock need of clock > 25MHz to run with USBC
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL1
```

Content of conf_clock.h for SAM3X and SAM3A devices (UOTGHS: USB OTG High Speed):

```
// USB Clock Source fixed at UPLL.
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_UPLL
#define CONFIG_USBCLK_DIV           1
```

Content of conf_clocks.h for SAMD21 devices (USB):

```
// USB Clock Source fixed at DFLL.
// SYSTEM_CLOCK_SOURCE_XOSC32K configuration - External 32KHz crystal/clock oscillator
#  define CONF_CLOCK_XOSC32K_ENABLE               true
#  define CONF_CLOCK_XOSC32K_EXTERNAL_CRYSTAL     SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC32K_STARTUP_TIME         SYSTEM_XOSC32K_STARTUP_65536
#  define CONF_CLOCK_XOSC32K_AUTO_AMPLITUDE_CONTROL  false
#  define CONF_CLOCK_XOSC32K_ENABLE_1KHZ_OUPUT    false
#  define CONF_CLOCK_XOSC32K_ENABLE_32KHZ_OUTPUT  true
#  define CONF_CLOCK_XOSC32K_ON_DEMAND            false
#  define CONF_CLOCK_XOSC32K_RUN_IN_STANDBY       true
// SYSTEM_CLOCK_SOURCE_DFLL configuration - Digital Frequency Locked Loop
#  define CONF_CLOCK_DFLL_ENABLE                  true
#  define CONF_CLOCK_DFLL_LOOP_MODE               SYSTEM_CLOCK_DFLL_LOOP_MODE_CLOSED
#  define CONF_CLOCK_DFLL_ON_DEMAND               true

// DFLL closed loop mode configuration
#  define CONF_CLOCK_DFLL_SOURCE_GCLK_GENERATOR   GCLK_GENERATOR_1
#  define CONF_CLOCK_DFLL_MULTIPLY_FACTOR         (48000000/32768)
#  define CONF_CLOCK_DFLL_QUICK_LOCK              true
#  define CONF_CLOCK_DFLL_TRACK_AFTER_FINE_LOCK   true
#  define CONF_CLOCK_DFLL_KEEP_LOCK_ON_WAKEUP     true
#  define CONF_CLOCK_DFLL_ENABLE_CHILL_CYCLE      true
#  define CONF_CLOCK_DFLL_MAX_COARSE_STEP_SIZE    (0x1f / 8)
#  define CONF_CLOCK_DFLL_MAX_FINE_STEP_SIZE      (0xff / 8)

#  define CONF_CLOCK_CONFIGURE_GCLK               true

// Configure GCLK generator 0 (Main Clock)
#  define CONF_CLOCK_GCLK_0_ENABLE                true
#  define CONF_CLOCK_GCLK_0_RUN_IN_STANDBY        true
#  define CONF_CLOCK_GCLK_0_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_DFLL
#  define CONF_CLOCK_GCLK_0_PRESCALER             1
#  define CONF_CLOCK_GCLK_0_OUTPUT_ENABLE         false

// Configure GCLK generator 1
#  define CONF_CLOCK_GCLK_1_ENABLE                true
#  define CONF_CLOCK_GCLK_1_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_1_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_XOSC32K
#  define CONF_CLOCK_GCLK_1_PRESCALER             1
#  define CONF_CLOCK_GCLK_1_OUTPUT_ENABLE         true
```

8.3 USB Host Advanced Use Cases

- Enable USB High Speed Support
- Multiple Classes Support
- Dual Roles Support

8.3.1 Enable USB High Speed Support

In this use case, the USB host is used to support USB high speed.

8.3.1.1 Setup Steps

Prior to implement this use case, be sure to have already applied the UHI module "basic use case".

8.3.1.2 Usage Steps

Example Code

Content of conf_usb_host.h:

```
#define USB_HOST_HS_SUPPORT
```

Workflow

1. Ensure that conf_usb_host.h is available and contains the following parameters required for a USB device high speed (480Mbit/s):

```
#define  USB_HOST_HS_SUPPORT
```

8.3.2 Multiple Classes Support

In this use case, the USB host is used to support several USB classes.

8.3.2.1 Setup Steps

Prior to implement this use case, be sure to have already applied the UHI module "basic use case".

8.3.2.2 Usage Steps

Example Code

Content of conf_usb_host.h:

```
#define USB_HOST_UHI   UHI_HID_MOUSE, UHI_MSC, UHI_CDC
```

Workflow

1. Ensure that conf_usb_host.h is available and contains the following parameters:

```
#define USB_HOST_UHI   UHI_HID_MOUSE, UHI_MSC, UHI_CDC
```

Note USB_HOST_UHI defines the list of UHI supported by USB host. Here, you must add all classes that you want to support.

8.3.3 Dual Roles Support

In this use case, the USB host and USB device are enabled, it is the dual role.

Note On the Atmel boards, the switch of USB role is managed automatically by the USB stack thank to a USB OTG connector and its USB ID pin. Refer to section "Dual roles" for further information in the application note:

- Atmel AVR4950: ASF - USB Host Stack[2]

8.3.3.1 Setup Steps

Prior to implement this use case, be sure to have already applied the UHI module "basic use case".

8.3.3.2 Usage Steps

Example Code

Content of conf_usb_host.h:

```
#define UHC_MODE_CHANGE(b_host_mode)   my_callback_mode_change(b_host_mode)
```

[2] http://www.atmel.com/images/doc8486.pdf

```
extern void my_callback_mode_change(bool b_host_mode);
```

Add to application C-file:

```
void usb_init(void)
{
  //udc_start();
  uhc_start();
}

bool my_host_mode;
void my_callback_mode_change(bool b_host_mode)
{
  my_host_mode = b_host_mode;
}

void my_usb_task(void)
{
  if (my_host_mode) {
    // CALL USB Host task
  } else {
    // CALL USB Device task
  }
}
```

Workflow

1. In case of USB dual roles (Device and Host), the USB stack must be enabled by uhc_start() and the udc_start() must not be called.

   ```
   //udc_start();
   uhc_start();
   ```

2. In dual role, to know the current USB mode, the callback to notify the mode changes can be used.

   - Ensure that conf_usb_host.h contains the following parameters:

     ```
     #define UHC_MODE_CHANGE(b_host_mode)   my_callback_mode_change(b_host_mode)
     extern void my_callback_mode_change(bool b_host_mode);
     ```

   - Ensure that application contains the following code:

     ```
     bool my_host_mode;
     void my_callback_mode_change(bool b_host_mode)
     {
       my_host_mode = b_host_mode;
     }

     void my_usb_task(void)
     {
       if (my_host_mode) {
         // CALL USB Host task
       } else {
         // CALL USB Device task
       }
     }
     ```

9. USB Host Interface (UHI) for Communication Class Device (CDC)

USB Host Interface (UHI) for Communication Class Device (CDC) provides an interface for the configuration and management of USB CDC serial host.

The outline of this documentation is as follows:

- API Overview
- Quick Start Guide for USB Host Communication Device Class Module (UHI CDC)
- Configuration File Examples

For more details for Atmel® Software Framework (ASF) USB Host Stack, refer to following application note:

- AVR4950: ASF - USB Host Stack[1]

9.1 API Overview

9.1.1 Macro Definitions

9.1.1.1 Interface with USB Host Core (UHC)

Definition and functions required by UHC.

Macro UHI_CDC

```
#define UHI_CDC \
  { \
 .install = uhi_cdc_install, \
 .enable = uhi_cdc_enable, \
 .uninstall = uhi_cdc_uninstall, \
 .sof_notify = uhi_cdc_sof, \
}
```

Global definition which contains standard UHI API for UHC. It must be added in USB_HOST_UHI define from conf_usb_host.h file.

9.1.2 Function Definitions

9.1.2.1 Functions Required by UHC

Function uhi_cdc_install()

Install interface Allocate interface endpoints if supported.

```
uhc_enum_status_t uhi_cdc_install(
  uhc_device_t * dev)
```

Table 9-1. Parameters

Data direction	Parameter name	Description
[in]	uhc_device_t	Device to request

[1] http://www.atmel.com/dyn/resources/prod_documents/doc8486.pdf

Returns Status of the install.

Function uhi_cdc_enable()

Enable the interface.

```
void uhi_cdc_enable(
  uhc_device_t * dev)
```

Enable a USB interface corresponding to UHI.

Table 9-2. Parameters

Data direction	Parameter name	Description
[in]	uhc_device_t	Device to request

Function uhi_cdc_uninstall()

Uninstall the interface (if installed).

```
void uhi_cdc_uninstall(
  uhc_device_t * dev)
```

Table 9-3. Parameters

Data direction	Parameter name	Description
[in]	uhc_device_t	Device to request

Function uhi_cdc_sof()

Signal that a SOF has occurred.

```
void uhi_cdc_sof(
  bool b_micro)
```

9.1.2.2 UHI for Communication Device Class

Common APIs used by high level application to use this USB host class. These routines are used by memory to transfer its data to/from USB CDC endpoint.

Function uhi_cdc_open()

Open a port of UHI CDC interface.

```
bool uhi_cdc_open(
```

```
  uint8_t port,
  usb_cdc_line_coding_t * configuration)
```

Table 9-4. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number
[in]	configuration	Pointer on port configuration

Returns `true` if the port is available.

Function uhi_cdc_close()

Close a port.

```
void uhi_cdc_close(
  uint8_t port)
```

Table 9-5. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number

Function uhi_cdc_is_rx_ready()

This function checks if a character has been received on the CDC line.

```
bool uhi_cdc_is_rx_ready(
  uint8_t port)
```

Table 9-6. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number

Returns `true` if a byte is ready to be read.

Function uhi_cdc_get_nb_received()

This function returns the number of character available on the CDC line.

```
iram_size_t uhi_cdc_get_nb_received(
  uint8_t port)
```

Table 9-7. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number

Returns The number of data received.

Function uhi_cdc_getc()

Waits and gets a value on CDC line.

```
int uhi_cdc_getc(
  uint8_t port)
```

Table 9-8. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number

Returns Value read on CDC line.

Function uhi_cdc_read_buf()

Reads a RAM buffer on CDC line.

```
iram_size_t uhi_cdc_read_buf(
  uint8_t port,
  void * buf,
  iram_size_t size)
```

Table 9-9. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number
[out]	buf	Values read
[in]	size	Number of value read

Returns The number of data remaining.

Function uhi_cdc_is_tx_ready()

This function checks if a new character sent is possible.

```
bool uhi_cdc_is_tx_ready(
```

```
  uint8_t port)
```

The type int is used to support scanf redirection from compiler LIB.

Table 9-10. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number

Returns `true` if a new character can be sent.

Function uhi_cdc_putc()

Puts a byte on CDC line The type int is used to support printf redirection from compiler LIB.

```
int uhi_cdc_putc(
  uint8_t port,
  int value)
```

Table 9-11. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number
[in]	value	Value to put

Returns `true` if function was successfully done, otherwise `false`.

Function uhi_cdc_write_buf()

Writes a RAM buffer on CDC line.

```
iram_size_t uhi_cdc_write_buf(
  uint8_t port,
  const void * buf,
  iram_size_t size)
```

Table 9-12. Parameters

Data direction	Parameter name	Description
[in]	port	Communication port number
[in]	buf	Values to write
[in]	size	Number of value to write

Returns The number of data remaining.

9.2 Quick Start Guide for USB Host Communication Device Class Module (UHI CDC)

This is the quick start guide for the USB Host Communication Device Class Module (UHI CDC) with step-by-step instructions on how to configure and use the modules in a selection of use cases.

The use cases highlights several code fragments. The code fragments in the steps for setup can be copied into a custom initialization function, while the steps for usage can be copied into, e.g., the main application function.

9.2.1 Basic Use Case

In this basic use case, the "USB Host CDC (Single Class support)" module is used.

The "USB Host CDC (Multiple Classes support)" module usage is described in Advanced Use Cases.

9.2.1.1 Setup Steps

As a USB host, it follows common USB host setup steps. Refer to USB Host Basic Setup.

9.2.1.2 Usage Steps

Example Code

Content of conf_usb_host.h:

```
#define USB_HOST_UHI        UHI_CDC
#define UHI_CDC_CHANGE(dev, b_plug) my_callback_cdc_change(dev, b_plug)
extern bool my_callback_cdc_change(uhc_device_t* dev, bool b_plug);
#define UHI_CDC_RX_NOTIFY() my_callback_cdc_rx_notify()
extern void my_callback_cdc_rx_notify(void);
#include "uhi_cdc.h" // At the end of conf_usb_host.h file
```

Add to application C-file:

```
static bool my_flag_cdc_available = false;
bool my_callback_cdc_change(uhc_device_t* dev, bool b_plug)
{
   if (b_plug) {

   // USB Device CDC connected
   my_flag_cdc_available = true;
   // Open and configure USB CDC ports
   usb_cdc_line_coding_t cfg = {
         .dwDTERate   = CPU_TO_LE32(115200),
         .bCharFormat = CDC_STOP_BITS_1,
         .bParityType = CDC_PAR_NONE,
         .bDataBits   = 8,
   };
   uhi_cdc_open(0, &cfg);

   } else {

   my_flag_cdc_available = false;

   }
}

void my_callback_cdc_rx_notify(void)
{
   // Wakeup my_task_rx() task
}

#define MESSAGE "Hello"
void my_task(void)
```

```
{
   static bool startup = true;

   if (!my_flag_cdc_available) {
      startup = true;
      return;
   }

   if (startup) {
      startup = false;
      // Send data on CDC communication port
      uhi_cdc_write_buf(0, MESSAGE, sizeof(MESSAGE)-1);
      uhi_cdc_putc(0,'\n');
      return;
   }
}

void my_task_rx(void)
{
   while (uhi_cdc_is_rx_ready(0)) {
      int value = uhi_cdc_getc(0);
   }
}
```

Workflow

1. Ensure that conf_usb_host.h is available and contains the following configuration which is the USB host CDC configuration:

```
#define USB_HOST_UHI   UHI_CDC
```

Note It defines the list of UHI supported by USB host.

```
#define UHI_CDC_CHANGE(dev, b_plug) my_callback_cdc_change(dev, b_plug)
extern bool my_callback_cdc_change(uhc_device_t* dev, bool b_plug);
```

Note This callback is called when a USB device CDC is plugged or unplugged. The communication port can be opened and configured here.

```
#define UHI_CDC_RX_NOTIFY() my_callback_cdc_rx_notify()
extern void my_callback_cdc_rx_notify(void);
```

Note This callback is called when a new data are received. This can be used to manage data reception through interrupt and avoid pooling.

2. The CDC data access functions are described in UHI CDC API Overview.

9.2.2 Advanced Use Cases

For more advanced use of the UHI CDC module, see the following:

- USB Host Advanced Use Cases

9.3 Configuration File Examples

9.3.1 conf_usb_host.h

9.3.1.1 UHI CDC Single

```
#ifndef _CONF_USB_HOST_H_
#define _CONF_USB_HOST_H_

#include "compiler.h"

#define USB_HOST_UHI        UHI_CDC

#define USB_HOST_POWER_MAX  500

// #define USB_HOST_HUB_SUPPORT

#if (UC3A3||UC3A4)
#  define USB_HOST_HS_SUPPORT
#endif

//#define UHC_MODE_CHANGE(b_host_mode)        usb_host_mode_change(b_host_mode)

//#define UHC_VBUS_CHANGE(b_present)          usb_host_vbus_change(b_present)

//#define UHC_VBUS_ERROR()                    usb_host_vbus_error()

//#define UHC_CONNECTION_EVENT(dev,b_present) usb_host_connection_event(dev,b_present)

//#define UHC_WAKEUP_EVENT()                  usb_host_wakeup_event()

//#define UHC_SOF_EVENT()                     usb_host_sof_event()

//#define UHC_DEVICE_CONF(dev)                uint8_t usb_host_device_conf(dev)

//#define UHC_ENUM_EVENT(dev,b_status)        usb_host_enum_event(dev,b_status)




#define UHI_CDC_CHANGE(dev,b_plug)
#define UHI_CDC_RX_NOTIFY()



#include "uhi_cdc.h"

#endif // _CONF_USB_HOST_H_
```

9.3.1.2 UHI CDC Multiple (Composite)

```
#ifndef _CONF_USB_HOST_H_
#define _CONF_USB_HOST_H_

#include "compiler.h"

#define USB_HOST_UHI        // UHI_MSC, UHI_HID_MOUSE, UHI_CDC, UHI_VENDOR
```

```
#define USB_HOST_POWER_MAX  500

// #define USB_HOST_HUB_SUPPORT

#if (UC3A3 || UC3A4)
#  define USB_HOST_HS_SUPPORT
#endif

//#define UHC_MODE_CHANGE(b_host_mode)        usb_host_mode_change(b_host_mode)

//#define UHC_VBUS_CHANGE(b_present)          usb_host_vbus_change(b_present)

//#define UHC_VBUS_ERROR()                    usb_host_vbus_error()

//#define UHC_CONNECTION_EVENT(dev,b_present) usb_host_connection_event(dev,b_present)

//#define UHC_WAKEUP_EVENT()                  usb_host_wakeup_event()

//#define UHC_SOF_EVENT()                     usb_host_sof_event()

//#define UHC_DEVICE_CONF(dev)                uint8_t usb_host_device_conf(dev)

//#define UHC_ENUM_EVENT(dev,b_status)        usb_host_enum_event(dev,b_status)




#define UHI_HID_MOUSE_CHANGE(dev,b_plug)
#define UHI_HID_MOUSE_EVENT_BTN_LEFT(b_state)
#define UHI_HID_MOUSE_EVENT_BTN_RIGHT(b_state)
#define UHI_HID_MOUSE_EVENT_BTN_MIDDLE(b_state)
#define UHI_HID_MOUSE_EVENT_MOUVE(x,y,scroll)


#define UHI_MSC_CHANGE(dev,b_plug)


#define UHI_CDC_CHANGE(dev,b_plug)
#define UHI_CDC_RX_NOTIFY()


#define UHI_VENDOR_CHANGE(dev, b_plug)
#define UHI_VENDOR_VID_PID_LIST {USB_VID_ATMEL, USB_PID_ATMEL_ASF_VENDOR_CLASS}




//#include "uhi_msc.h"
//#include "uhi_hid_mouse.h"

#endif // _CONF_USB_HOST_H_
```

9.3.2 conf_clock.h

9.3.2.1 AT32UC3A0, AT32UC3A1, AT32UC3B Devices (USBB)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED
```

```
// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL0

// ===== PLL0 Options
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_MUL             4 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL0_DIV             1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
#define CONFIG_PLL1_SOURCE            PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
#define CONFIG_PLL1_MUL               8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL1_DIV               2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
//#define CONFIG_SYSCLK_CPU_DIV         0 /* Fcpu = Fsys/(2 ^ CPU_div) */
//#define CONFIG_SYSCLK_PBA_DIV         0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV         0 /* Fpbb = Fsys/(2 ^ PBB_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

9.3.2.2 AT32UC3A3, AT32UC3A4 Devices (USBB with High Speed Support)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL0

// ===== PLL0 Options
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_MUL             11 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL0_DIV             2  /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL1_MUL             8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL1_DIV             2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
```

```
#define CONFIG_SYSCLK_CPU_DIV         0 /* Fcpu = Fsys/(2 ^ CPU_div) */
#define CONFIG_SYSCLK_PBA_DIV         0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV       0 /* Fpbb = Fsys/(2 ^ PBB_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

9.3.2.3 AT32UC3C, ATUCXXD, ATUCXXL3U, ATUCXXL4U Devices (USBC)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC1
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_PLL0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL1
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RC8M

// ===== PLL0 Options
#define CONFIG_PLL0_SOURCE            PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_SOURCE          PLL_SRC_RC8M
#define CONFIG_PLL0_MUL               3 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL0_DIV               1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL1_SOURCE          PLL_SRC_RC8M
//#define CONFIG_PLL1_MUL             3 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL1_DIV             1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
//#define CONFIG_SYSCLK_CPU_DIV       0 /* Fcpu = Fsys/(2 ^ CPU_div) */
//#define CONFIG_SYSCLK_PBA_DIV       0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV       0 /* Fpbb = Fsys/(2 ^ PBB_div) */
//#define CONFIG_SYSCLK_PBC_DIV       0 /* Fpbc = Fsys/(2 ^ PBC_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC1
```

```
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

9.3.2.4 SAM3X, SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock (MCK) Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_BYPASS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_4M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_8M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_12M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_BYPASS
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLLACK
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_UPLLCK

// ===== System Clock (MCK) Prescaler Options   (Fmck = Fsys / (SYSCLK_PRES))
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_1
#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_2
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_4
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_8
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_16
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_32
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_64
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_3

// ===== PLL0 (A) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
// Use mul and div effective values here.
#define CONFIG_PLL0_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL0_MUL             14
#define CONFIG_PLL0_DIV             1

// ===== UPLL (UTMI) Hardware fixed at 480MHz.

// ===== USB Clock Source Options   (Fusb = FpllX / USB_div)
// Use div effective value here.
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_UPLL
#define CONFIG_USBCLK_DIV           1

// ===== Target frequency (System clock)
// - XTAL frequency: 12MHz
// - System clock source: PLLA
// - System clock prescaler: 2 (divided by 2)
// - PLLA source: XTAL
// - PLLA output: XTAL * 14 / 1
// - System clock is: 12 * 14 / 1 /2 = 84MHz
// ===== Target frequency (USB Clock)
// - USB clock source: UPLL
// - USB clock divider: 1 (not divided)
// - UPLL frequency: 480MHz
// - USB clock: 480 / 1 = 480MHz
```

```
#endif /* CONF_CLOCK_H_INCLUDED */
```

9.3.3 conf_clocks.h

9.3.3.1 SAMD21 Devices (USB)

```
#include <clock.h>

#ifndef CONF_CLOCKS_H_INCLUDED
#  define CONF_CLOCKS_H_INCLUDED

/* System clock bus configuration */
#  define CONF_CLOCK_CPU_CLOCK_FAILURE_DETECT     false
#  define CONF_CLOCK_FLASH_WAIT_STATES            2
#  define CONF_CLOCK_CPU_DIVIDER                  SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBA_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBB_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1

/* SYSTEM_CLOCK_SOURCE_OSC8M configuration - Internal 8MHz oscillator */
#  define CONF_CLOCK_OSC8M_PRESCALER              SYSTEM_OSC8M_DIV_1
#  define CONF_CLOCK_OSC8M_ON_DEMAND              true
#  define CONF_CLOCK_OSC8M_RUN_IN_STANDBY         false

/* SYSTEM_CLOCK_SOURCE_XOSC configuration - External clock/oscillator */
#  define CONF_CLOCK_XOSC_ENABLE                  false
#  define CONF_CLOCK_XOSC_EXTERNAL_CRYSTAL        SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC_EXTERNAL_FREQUENCY      12000000UL
#  define CONF_CLOCK_XOSC_STARTUP_TIME            SYSTEM_XOSC_STARTUP_32768
#  define CONF_CLOCK_XOSC_AUTO_GAIN_CONTROL       true
#  define CONF_CLOCK_XOSC_ON_DEMAND               true
#  define CONF_CLOCK_XOSC_RUN_IN_STANDBY          false

/* SYSTEM_CLOCK_SOURCE_XOSC32K configuration - External 32KHz crystal/clock oscillator */
#  define CONF_CLOCK_XOSC32K_ENABLE               false
#  define CONF_CLOCK_XOSC32K_EXTERNAL_CRYSTAL     SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC32K_STARTUP_TIME         SYSTEM_XOSC32K_STARTUP_65536
#  define CONF_CLOCK_XOSC32K_AUTO_AMPLITUDE_CONTROL  false
#  define CONF_CLOCK_XOSC32K_ENABLE_1KHZ_OUPUT    false
#  define CONF_CLOCK_XOSC32K_ENABLE_32KHZ_OUTPUT  true
#  define CONF_CLOCK_XOSC32K_ON_DEMAND            true
#  define CONF_CLOCK_XOSC32K_RUN_IN_STANDBY       false

/* SYSTEM_CLOCK_SOURCE_OSC32K configuration - Internal 32KHz oscillator */
#  define CONF_CLOCK_OSC32K_ENABLE                false
#  define CONF_CLOCK_OSC32K_STARTUP_TIME          SYSTEM_OSC32K_STARTUP_130
#  define CONF_CLOCK_OSC32K_ENABLE_1KHZ_OUTPUT    true
#  define CONF_CLOCK_OSC32K_ENABLE_32KHZ_OUTPUT   true
#  define CONF_CLOCK_OSC32K_ON_DEMAND             true
#  define CONF_CLOCK_OSC32K_RUN_IN_STANDBY        false

/* SYSTEM_CLOCK_SOURCE_DFLL configuration - Digital Frequency Locked Loop */
#  define CONF_CLOCK_DFLL_ENABLE                  true
#  define CONF_CLOCK_DFLL_LOOP_MODE               SYSTEM_CLOCK_DFLL_LOOP_MODE_USB_RECOVERY
#  define CONF_CLOCK_DFLL_ON_DEMAND               true

/* DFLL open loop mode configuration */
#  define CONF_CLOCK_DFLL_COARSE_VALUE            (0x1f / 4)
#  define CONF_CLOCK_DFLL_FINE_VALUE              (0xff / 4)
```

```
/* DFLL closed loop mode configuration */
#  define CONF_CLOCK_DFLL_SOURCE_GCLK_GENERATOR   GCLK_GENERATOR_1
#  define CONF_CLOCK_DFLL_MULTIPLY_FACTOR         (48000000 / 32768)
#  define CONF_CLOCK_DFLL_QUICK_LOCK              true
#  define CONF_CLOCK_DFLL_TRACK_AFTER_FINE_LOCK   true
#  define CONF_CLOCK_DFLL_KEEP_LOCK_ON_WAKEUP     true
#  define CONF_CLOCK_DFLL_ENABLE_CHILL_CYCLE      true
#  define CONF_CLOCK_DFLL_MAX_COARSE_STEP_SIZE    (0x1f / 4)
#  define CONF_CLOCK_DFLL_MAX_FINE_STEP_SIZE      (0xff / 4)

/* SYSTEM_CLOCK_SOURCE_DPLL configuration - Digital Phase-Locked Loop */
#  define CONF_CLOCK_DPLL_ENABLE                  false
#  define CONF_CLOCK_DPLL_ON_DEMAND               true
#  define CONF_CLOCK_DPLL_RUN_IN_STANDBY          false
#  define CONF_CLOCK_DPLL_LOCK_BYPASS             false
#  define CONF_CLOCK_DPLL_WAKE_UP_FAST            false
#  define CONF_CLOCK_DPLL_LOW_POWER_ENABLE        false

#  define CONF_CLOCK_DPLL_LOCK_TIME               SYSTEM_CLOCK_SOURCE_DPLL_LOCK_TIME_NO_TIMEOUT
#  define CONF_CLOCK_DPLL_REFERENCE_CLOCK         SYSTEM_CLOCK_SOURCE_DPLL_REFERENCE_CLOCK_REF0
#  define CONF_CLOCK_DPLL_FILTER                  SYSTEM_CLOCK_SOURCE_DPLL_FILTER_DEFAULT

#  define CONF_CLOCK_DPLL_REFERENCE_FREQUENCY     32768
#  define CONF_CLOCK_DPLL_REFEREMCE_DIVIDER       1
#  define CONF_CLOCK_DPLL_OUTPUT_FREQUENCY        48000000

/* Set this to true to configure the GCLK when running clocks_init. If set to
 * false, none of the GCLK generators will be configured in clocks_init(). */
#  define CONF_CLOCK_CONFIGURE_GCLK               true

/* Configure GCLK generator 0 (Main Clock) */
#  define CONF_CLOCK_GCLK_0_ENABLE                true
#  define CONF_CLOCK_GCLK_0_RUN_IN_STANDBY        true
#  define CONF_CLOCK_GCLK_0_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_DFLL
#  define CONF_CLOCK_GCLK_0_PRESCALER             1
#  define CONF_CLOCK_GCLK_0_OUTPUT_ENABLE         false

/* Configure GCLK generator 1 */
#  define CONF_CLOCK_GCLK_1_ENABLE                false
#  define CONF_CLOCK_GCLK_1_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_1_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_XOSC32K
#  define CONF_CLOCK_GCLK_1_PRESCALER             1
#  define CONF_CLOCK_GCLK_1_OUTPUT_ENABLE         false

/* Configure GCLK generator 2 (RTC) */
#  define CONF_CLOCK_GCLK_2_ENABLE                false
#  define CONF_CLOCK_GCLK_2_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_2_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC32K
#  define CONF_CLOCK_GCLK_2_PRESCALER             32
#  define CONF_CLOCK_GCLK_2_OUTPUT_ENABLE         false

/* Configure GCLK generator 3 */
#  define CONF_CLOCK_GCLK_3_ENABLE                false
#  define CONF_CLOCK_GCLK_3_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_3_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_3_PRESCALER             1
#  define CONF_CLOCK_GCLK_3_OUTPUT_ENABLE         false

/* Configure GCLK generator 4 */
#  define CONF_CLOCK_GCLK_4_ENABLE                false
#  define CONF_CLOCK_GCLK_4_RUN_IN_STANDBY        false
```

```
#  define CONF_CLOCK_GCLK_4_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_4_PRESCALER             1
#  define CONF_CLOCK_GCLK_4_OUTPUT_ENABLE         false

/* Configure GCLK generator 5 */
#  define CONF_CLOCK_GCLK_5_ENABLE                false
#  define CONF_CLOCK_GCLK_5_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_5_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_5_PRESCALER             1
#  define CONF_CLOCK_GCLK_5_OUTPUT_ENABLE         false

/* Configure GCLK generator 6 */
#  define CONF_CLOCK_GCLK_6_ENABLE                false
#  define CONF_CLOCK_GCLK_6_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_6_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_6_PRESCALER             1
#  define CONF_CLOCK_GCLK_6_OUTPUT_ENABLE         false

/* Configure GCLK generator 7 */
#  define CONF_CLOCK_GCLK_7_ENABLE                false
#  define CONF_CLOCK_GCLK_7_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_7_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_7_PRESCALER             1
#  define CONF_CLOCK_GCLK_7_OUTPUT_ENABLE         false

#endif /* CONF_CLOCKS_H_INCLUDED */
```

9.3.4 conf_board.h

9.3.4.1 AT32UC3A0, AT32UC3A1, AT32UC3B Devices (USBB)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

9.3.4.2 AT32UC3A3, AT32UC3A4 Devices (USBB with High Speed Support)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

9.3.4.3 AT32UC3C, ATUCXXD, ATUCXXL3U, ATUCXXL4U Devices (USBC)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

9.3.4.4 SAM3X, SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// USB pins are used
#define CONF_BOARD_USB_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

9.3.4.5 SAMD21 Devices (USB)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

/* Enable USB VBUS detect */
#define CONF_BOARD_USB_VBUS_DETECT

#endif /* CONF_BOARD_H_INCLUDED */
```

10. USB Host Interface (UHI) for Human Interface Device Mouse (HID Mouse)

USB Host Interface (UHI) for Human Interface Device Mouse (HID Mouse) provides an interface for the configuration and management of USB HID mouse host.

The outline of this documentation is as follows:

- API Overview
- Quick Start Guide for USB Host Mouse Module (UHI Mouse)
- Configuration File Examples

For more details for Atmel® Software Framework (ASF) USB Host Stack, refer to following application note:

- AVR4950: ASF - USB Host Stack[1]

10.1 API Overview

10.1.1 Macro Definitions

10.1.1.1 Interface with USB Host Core (UHC)

Define and functions required by UHC.

Macro UHI_HID_MOUSE

```
#define UHI_HID_MOUSE \
  { \
 .install = uhi_hid_mouse_install, \
 .enable = uhi_hid_mouse_enable, \
 .uninstall = uhi_hid_mouse_uninstall, \
 .sof_notify = NULL, \
}
```

Global define which contains standard UHI API for UHC.

It must be added in USB_HOST_UHI define from conf_usb_host.h file.

10.1.1.2 UHI for Human Interface Device Mouse Class

Common APIs used by high level application to use this USB host class.

These APIs require only callback definitions in conf_usb_host.h file through following defines:

Macro UHI_HID_MOUSE_CHANGE

```
#define UHI_HID_MOUSE_CHANGE(dev, b_plug) \
```

Macro UHI_HID_MOUSE_EVENT_BTN_LEFT

```
#define UHI_HID_MOUSE_EVENT_BTN_LEFT(b_state) \
```

[1] http://www.atmel.com/dyn/resources/prod_documents/doc8486.pdf

Macro UHI_HID_MOUSE_EVENT_BTN_RIGHT

```
#define UHI_HID_MOUSE_EVENT_BTN_RIGHT(b_state) \
```

Macro UHI_HID_MOUSE_EVENT_BTN_MIDDLE

```
#define UHI_HID_MOUSE_EVENT_BTN_MIDDLE(b_state) \
```

Macro UHI_HID_MOUSE_EVENT_MOUVE

```
#define UHI_HID_MOUSE_EVENT_MOUVE(x, y, scroll) \
```

10.1.2 Function Definitions

10.1.2.1 Functions Required by UHC

Function uhi_hid_mouse_install()

Install interface Allocate interface endpoints if supported.

```
uhc_enum_status_t uhi_hid_mouse_install(
  uhc_device_t * dev)
```

Table 10-1. Parameters

Data direction	Parameter name	Description
[in]	uhc_device_t	Device to request

Returns

Status of the install.

Function uhi_hid_mouse_enable()

Enable the interface.

```
void uhi_hid_mouse_enable(
  uhc_device_t * dev)
```

Enable a USB interface corresponding to UHI.

Table 10-2. Parameters

Data direction	Parameter name	Description
[in]	uhc_device_t	Device to request

Function uhi_hid_mouse_uninstall()

Uninstall the interface (if installed).

```
void uhi_hid_mouse_uninstall(
  uhc_device_t * dev)
```

Table 10-3. Parameters

Data direction	Parameter name	Description
[in]	uhc_device_t	Device to request

10.2 Quick Start Guide for USB Host Mouse Module (UHI Mouse)

This is the quick start guide for the USB Host Mouse Module (UHI Mouse) with step-by-step instructions on how to configure and use the modules in a selection of use cases.

The use cases highlights several code fragments. The code fragments in the steps for setup can be copied into a custom initialization function, while the steps for usage can be copied into, e.g., the main application function.

10.2.1 Basic Use Case

In this basic use case, the "USB Host HID Mouse (Single Class support)" module is used. The "USB Host HID Mouse (Multiple Classes support)" module usage is described in Advanced Use Cases.

10.2.1.1 Setup Steps

As a USB host, it follows common USB host setup steps. Refer to USB Host Basic Setup.

10.2.1.2 Usage Steps

Example Code

Content of conf_usb_host.h:

```
#define USB_HOST_UHI        UHI_HID_MOUSE
#define UHI_HID_MOUSE_CHANGE(dev, b_plug) my_callback_mouse_change(dev, b_plug)
extern bool my_callback_mouse_change(uhc_device_t* dev, bool b_plug);
#define UHI_HID_MOUSE_EVENT_BTN_LEFT(b_state) my_callback_event_btn_left(b_state)
extern void my_callback_event_btn_left(bool b_state);
#define UHI_HID_MOUSE_EVENT_BTN_RIGHT(b_state) my_callback_event_btn_right(b_state)
extern void my_callback_event_btn_right(bool b_state);
#define UHI_HID_MOUSE_EVENT_BTN_MIDDLE(b_state) my_callback_event_btn_middle(b_state)
extern void my_callback_event_btn_middle(bool b_state);
#define UHI_HID_MOUSE_EVENT_MOUVE(x, y, scroll) my_callback_event_mouse(x, y, scroll)
extern void my_callback_event_mouse(int8_t x, int8_t y, int8_t scroll);
#include "uhi_hid_mouse.h" // At the end of conf_usb_host.h file
```

Add to application C-file:

```
bool my_callback_mouse_change(uhc_device_t* dev, bool b_plug)
{
   if (b_plug) {
```

```
      my_display_on_mouse_icon();
   } else {
      my_display_off_mouse_icon();
   }
}

void my_callback_event_btn_left(bool b_state)
{
   if (b_state) {
      // Here mouse button left pressed
   } else {
      // Here mouse button left released
   }
}

void my_callback_event_mouse(int8_t x, int8_t y, int8_t scroll)
{
   if (!x) {
      // Here mouse are moved on axe X
      cursor_x += x;
   }
   if (!y) {
      // Here mouse are moved on axe Y
      cursor_y += y;
   }
   if (!scroll) {
      // Here mouse are moved the wheel
      wheel += scroll;
   }
}
```

Workflow

1. Ensure that conf_usb_host.h is available and contains the following configuration which is the USB host mouse configuration:

```
#define USB_HOST_UHI   UHI_HID_MOUSE
```

Note It defines the list of UHI supported by USB host.

```
#define UHI_HID_MOUSE_CHANGE(dev, b_plug) my_callback_mouse_change(dev, b_plug)
extern bool my_callback_mouse_change(uhc_device_t* dev, bool b_plug);
```

Note This callback is called when a USB device mouse is plugged or unplugged.

```
#define UHI_HID_MOUSE_EVENT_BTN_LEFT(b_state) my_callback_event_btn_left(b_state)
extern void my_callback_event_btn_left(bool b_state);
#define UHI_HID_MOUSE_EVENT_BTN_RIGHT(b_state) my_callback_event_btn_right(b_state)
extern void my_callback_event_btn_right(bool b_state);
#define UHI_HID_MOUSE_EVENT_BTN_MIDDLE(b_state) my_callback_event_btn_middle(b_state)
extern void my_callback_event_btn_middle(bool b_state);
#define UHI_HID_MOUSE_EVENT_MOUVE(x, y, scroll) my_callback_event_mouse(x, y, scroll)
extern void my_callback_event_mouse(int8_t x, int8_t y, int8_t scroll)
```

Note These callbacks are called when a USB device mouse event is received.

10.2.2 Advanced Use Cases

For more advanced use of the UHI HID mouse module, see the following:

- USB Host Advanced Use Cases

10.3 Configuration File Examples

10.3.1 conf_usb_host.h

10.3.1.1 UHI HID MOUSE Single

```
#ifndef _CONF_USB_HOST_H_
#define _CONF_USB_HOST_H_

#include "compiler.h"

#define USB_HOST_UHI        UHI_CDC

#define USB_HOST_POWER_MAX  500

// #define USB_HOST_HUB_SUPPORT

#if (UC3A3||UC3A4)
#  define USB_HOST_HS_SUPPORT
#endif

//#define UHC_MODE_CHANGE(b_host_mode)        usb_host_mode_change(b_host_mode)

//#define UHC_VBUS_CHANGE(b_present)          usb_host_vbus_change(b_present)

//#define UHC_VBUS_ERROR()                    usb_host_vbus_error()

//#define UHC_CONNECTION_EVENT(dev,b_present) usb_host_connection_event(dev,b_present)

//#define UHC_WAKEUP_EVENT()                  usb_host_wakeup_event()

//#define UHC_SOF_EVENT()                     usb_host_sof_event()

//#define UHC_DEVICE_CONF(dev)                uint8_t usb_host_device_conf(dev)

//#define UHC_ENUM_EVENT(dev,b_status)        usb_host_enum_event(dev,b_status)




#define UHI_CDC_CHANGE(dev,b_plug)
#define UHI_CDC_RX_NOTIFY()



#include "uhi_cdc.h"

#endif // _CONF_USB_HOST_H_
```

10.3.1.2 UHI HID MOUSE Multiple (Composite)

```
#ifndef _CONF_USB_HOST_H_
#define _CONF_USB_HOST_H_

#include "compiler.h"

#define USB_HOST_UHI        // UHI_MSC, UHI_HID_MOUSE, UHI_CDC, UHI_VENDOR

#define USB_HOST_POWER_MAX  500

// #define USB_HOST_HUB_SUPPORT

#if (UC3A3 || UC3A4)
#  define USB_HOST_HS_SUPPORT
#endif

//#define UHC_MODE_CHANGE(b_host_mode)        usb_host_mode_change(b_host_mode)

//#define UHC_VBUS_CHANGE(b_present)          usb_host_vbus_change(b_present)

//#define UHC_VBUS_ERROR()                    usb_host_vbus_error()

//#define UHC_CONNECTION_EVENT(dev,b_present) usb_host_connection_event(dev,b_present)

//#define UHC_WAKEUP_EVENT()                  usb_host_wakeup_event()

//#define UHC_SOF_EVENT()                     usb_host_sof_event()

//#define UHC_DEVICE_CONF(dev)                uint8_t usb_host_device_conf(dev)

//#define UHC_ENUM_EVENT(dev,b_status)        usb_host_enum_event(dev,b_status)




#define UHI_HID_MOUSE_CHANGE(dev,b_plug)
#define UHI_HID_MOUSE_EVENT_BTN_LEFT(b_state)
#define UHI_HID_MOUSE_EVENT_BTN_RIGHT(b_state)
#define UHI_HID_MOUSE_EVENT_BTN_MIDDLE(b_state)
#define UHI_HID_MOUSE_EVENT_MOUVE(x,y,scroll)


#define UHI_MSC_CHANGE(dev,b_plug)


#define UHI_CDC_CHANGE(dev,b_plug)
#define UHI_CDC_RX_NOTIFY()


#define UHI_VENDOR_CHANGE(dev, b_plug)
#define UHI_VENDOR_VID_PID_LIST {USB_VID_ATMEL, USB_PID_ATMEL_ASF_VENDOR_CLASS}




//#include "uhi_msc.h"
//#include "uhi_hid_mouse.h"
```

```
#endif // _CONF_USB_HOST_H_
```

10.3.2 conf_clock.h

10.3.2.1 AT32UC3A0, AT32UC3A1, AT32UC3B Devices (USBB)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL0

// ===== PLL0 Options
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_MUL             4 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL0_DIV             1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
#define CONFIG_PLL1_SOURCE            PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
#define CONFIG_PLL1_MUL               8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL1_DIV               2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
//#define CONFIG_SYSCLK_CPU_DIV         0 /* Fcpu = Fsys/(2 ^ CPU_div) */
//#define CONFIG_SYSCLK_PBA_DIV         0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV         0 /* Fpbb = Fsys/(2 ^ PBB_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

10.3.2.2 AT32UC3A3, AT32UC3A4 Devices (USBB with High Speed Support)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL0

// ===== PLL0 Options
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_MUL             11 /* Fpll = (Fclk * PLL_mul) / PLL_div */
```

```
//#define CONFIG_PLL0_DIV             2  /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL1_MUL             8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL1_DIV             2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
#define CONFIG_SYSCLK_CPU_DIV         0 /* Fcpu = Fsys/(2 ^ CPU_div) */
#define CONFIG_SYSCLK_PBA_DIV         0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV       0 /* Fpbb = Fsys/(2 ^ PBB_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

10.3.2.3 AT32UC3C, ATUCXXD, ATUCXXL3U, ATUCXXL4U Devices (USBC)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC1
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_PLL0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL1
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RC8M

// ===== PLL0 Options
#define CONFIG_PLL0_SOURCE            PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_SOURCE          PLL_SRC_RC8M
#define CONFIG_PLL0_MUL               3 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL0_DIV               1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL1_SOURCE          PLL_SRC_RC8M
//#define CONFIG_PLL1_MUL             3 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL1_DIV             1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
//#define CONFIG_SYSCLK_CPU_DIV       0 /* Fcpu = Fsys/(2 ^ CPU_div) */
//#define CONFIG_SYSCLK_PBA_DIV       0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV       0 /* Fpbb = Fsys/(2 ^ PBB_div) */
//#define CONFIG_SYSCLK_PBC_DIV       0 /* Fpbc = Fsys/(2 ^ PBC_div) */
```

```
// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC1
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

10.3.2.4 SAM3X, SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock (MCK) Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_BYPASS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_4M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_8M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_12M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_BYPASS
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLLACK
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_UPLLCK

// ===== System Clock (MCK) Prescaler Options   (Fmck = Fsys / (SYSCLK_PRES))
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_1
#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_2
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_4
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_8
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_16
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_32
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_64
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_3

// ===== PLL0 (A) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
// Use mul and div effective values here.
#define CONFIG_PLL0_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL0_MUL             14
#define CONFIG_PLL0_DIV             1

// ===== UPLL (UTMI) Hardware fixed at 480MHz.

// ===== USB Clock Source Options   (Fusb = FpllX / USB_div)
// Use div effective value here.
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_UPLL
#define CONFIG_USBCLK_DIV           1

// ===== Target frequency (System clock)
// - XTAL frequency: 12MHz
// - System clock source: PLLA
// - System clock prescaler: 2 (divided by 2)
```

```
// - PLLA source: XTAL
// - PLLA output: XTAL * 14 / 1
// - System clock is: 12 * 14 / 1 /2 = 84MHz
// ===== Target frequency (USB Clock)
// - USB clock source: UPLL
// - USB clock divider: 1 (not divided)
// - UPLL frequency: 480MHz
// - USB clock: 480 / 1 = 480MHz


#endif /* CONF_CLOCK_H_INCLUDED */
```

10.3.3 conf_clocks.h

10.3.3.1 SAMD21 Devices (USB)

```
#include <clock.h>

#ifndef CONF_CLOCKS_H_INCLUDED
#  define CONF_CLOCKS_H_INCLUDED

/* System clock bus configuration */
#  define CONF_CLOCK_CPU_CLOCK_FAILURE_DETECT     false
#  define CONF_CLOCK_FLASH_WAIT_STATES            2
#  define CONF_CLOCK_CPU_DIVIDER                  SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBA_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBB_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1

/* SYSTEM_CLOCK_SOURCE_OSC8M configuration - Internal 8MHz oscillator */
#  define CONF_CLOCK_OSC8M_PRESCALER              SYSTEM_OSC8M_DIV_1
#  define CONF_CLOCK_OSC8M_ON_DEMAND              true
#  define CONF_CLOCK_OSC8M_RUN_IN_STANDBY         false

/* SYSTEM_CLOCK_SOURCE_XOSC configuration - External clock/oscillator */
#  define CONF_CLOCK_XOSC_ENABLE                  false
#  define CONF_CLOCK_XOSC_EXTERNAL_CRYSTAL        SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC_EXTERNAL_FREQUENCY      12000000UL
#  define CONF_CLOCK_XOSC_STARTUP_TIME            SYSTEM_XOSC_STARTUP_32768
#  define CONF_CLOCK_XOSC_AUTO_GAIN_CONTROL       true
#  define CONF_CLOCK_XOSC_ON_DEMAND               true
#  define CONF_CLOCK_XOSC_RUN_IN_STANDBY          false

/* SYSTEM_CLOCK_SOURCE_XOSC32K configuration - External 32KHz crystal/clock oscillator */
#  define CONF_CLOCK_XOSC32K_ENABLE               false
#  define CONF_CLOCK_XOSC32K_EXTERNAL_CRYSTAL     SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC32K_STARTUP_TIME         SYSTEM_XOSC32K_STARTUP_65536
#  define CONF_CLOCK_XOSC32K_AUTO_AMPLITUDE_CONTROL  false
#  define CONF_CLOCK_XOSC32K_ENABLE_1KHZ_OUPUT    false
#  define CONF_CLOCK_XOSC32K_ENABLE_32KHZ_OUTPUT  true
#  define CONF_CLOCK_XOSC32K_ON_DEMAND            true
#  define CONF_CLOCK_XOSC32K_RUN_IN_STANDBY       false

/* SYSTEM_CLOCK_SOURCE_OSC32K configuration - Internal 32KHz oscillator */
#  define CONF_CLOCK_OSC32K_ENABLE                false
#  define CONF_CLOCK_OSC32K_STARTUP_TIME          SYSTEM_OSC32K_STARTUP_130
#  define CONF_CLOCK_OSC32K_ENABLE_1KHZ_OUTPUT    true
#  define CONF_CLOCK_OSC32K_ENABLE_32KHZ_OUTPUT   true
#  define CONF_CLOCK_OSC32K_ON_DEMAND             true
#  define CONF_CLOCK_OSC32K_RUN_IN_STANDBY        false
```

```
/* SYSTEM_CLOCK_SOURCE_DFLL configuration - Digital Frequency Locked Loop */
#  define CONF_CLOCK_DFLL_ENABLE                   true
#  define CONF_CLOCK_DFLL_LOOP_MODE                SYSTEM_CLOCK_DFLL_LOOP_MODE_USB_RECOVERY
#  define CONF_CLOCK_DFLL_ON_DEMAND                true

/* DFLL open loop mode configuration */
#  define CONF_CLOCK_DFLL_COARSE_VALUE             (0x1f / 4)
#  define CONF_CLOCK_DFLL_FINE_VALUE               (0xff / 4)

/* DFLL closed loop mode configuration */
#  define CONF_CLOCK_DFLL_SOURCE_GCLK_GENERATOR    GCLK_GENERATOR_1
#  define CONF_CLOCK_DFLL_MULTIPLY_FACTOR          (48000000 / 32768)
#  define CONF_CLOCK_DFLL_QUICK_LOCK               true
#  define CONF_CLOCK_DFLL_TRACK_AFTER_FINE_LOCK    true
#  define CONF_CLOCK_DFLL_KEEP_LOCK_ON_WAKEUP      true
#  define CONF_CLOCK_DFLL_ENABLE_CHILL_CYCLE       true
#  define CONF_CLOCK_DFLL_MAX_COARSE_STEP_SIZE     (0x1f / 4)
#  define CONF_CLOCK_DFLL_MAX_FINE_STEP_SIZE       (0xff / 4)

/* SYSTEM_CLOCK_SOURCE_DPLL configuration - Digital Phase-Locked Loop */
#  define CONF_CLOCK_DPLL_ENABLE                   false
#  define CONF_CLOCK_DPLL_ON_DEMAND                true
#  define CONF_CLOCK_DPLL_RUN_IN_STANDBY           false
#  define CONF_CLOCK_DPLL_LOCK_BYPASS              false
#  define CONF_CLOCK_DPLL_WAKE_UP_FAST             false
#  define CONF_CLOCK_DPLL_LOW_POWER_ENABLE         false

#  define CONF_CLOCK_DPLL_LOCK_TIME                SYSTEM_CLOCK_SOURCE_DPLL_LOCK_TIME_NO_TIMEOUT
#  define CONF_CLOCK_DPLL_REFERENCE_CLOCK          SYSTEM_CLOCK_SOURCE_DPLL_REFERENCE_CLOCK_REF0
#  define CONF_CLOCK_DPLL_FILTER                   SYSTEM_CLOCK_SOURCE_DPLL_FILTER_DEFAULT

#  define CONF_CLOCK_DPLL_REFERENCE_FREQUENCY      32768
#  define CONF_CLOCK_DPLL_REFEREMCE_DIVIDER        1
#  define CONF_CLOCK_DPLL_OUTPUT_FREQUENCY         48000000

/* Set this to true to configure the GCLK when running clocks_init. If set to
 * false, none of the GCLK generators will be configured in clocks_init(). */
#  define CONF_CLOCK_CONFIGURE_GCLK                true

/* Configure GCLK generator 0 (Main Clock) */
#  define CONF_CLOCK_GCLK_0_ENABLE                 true
#  define CONF_CLOCK_GCLK_0_RUN_IN_STANDBY         true
#  define CONF_CLOCK_GCLK_0_CLOCK_SOURCE           SYSTEM_CLOCK_SOURCE_DFLL
#  define CONF_CLOCK_GCLK_0_PRESCALER              1
#  define CONF_CLOCK_GCLK_0_OUTPUT_ENABLE          false

/* Configure GCLK generator 1 */
#  define CONF_CLOCK_GCLK_1_ENABLE                 false
#  define CONF_CLOCK_GCLK_1_RUN_IN_STANDBY         false
#  define CONF_CLOCK_GCLK_1_CLOCK_SOURCE           SYSTEM_CLOCK_SOURCE_XOSC32K
#  define CONF_CLOCK_GCLK_1_PRESCALER              1
#  define CONF_CLOCK_GCLK_1_OUTPUT_ENABLE          false

/* Configure GCLK generator 2 (RTC) */
#  define CONF_CLOCK_GCLK_2_ENABLE                 false
#  define CONF_CLOCK_GCLK_2_RUN_IN_STANDBY         false
#  define CONF_CLOCK_GCLK_2_CLOCK_SOURCE           SYSTEM_CLOCK_SOURCE_OSC32K
#  define CONF_CLOCK_GCLK_2_PRESCALER              32
#  define CONF_CLOCK_GCLK_2_OUTPUT_ENABLE          false

/* Configure GCLK generator 3 */
```

```
#  define CONF_CLOCK_GCLK_3_ENABLE                false
#  define CONF_CLOCK_GCLK_3_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_3_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_3_PRESCALER             1
#  define CONF_CLOCK_GCLK_3_OUTPUT_ENABLE         false

/* Configure GCLK generator 4 */
#  define CONF_CLOCK_GCLK_4_ENABLE                false
#  define CONF_CLOCK_GCLK_4_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_4_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_4_PRESCALER             1
#  define CONF_CLOCK_GCLK_4_OUTPUT_ENABLE         false

/* Configure GCLK generator 5 */
#  define CONF_CLOCK_GCLK_5_ENABLE                false
#  define CONF_CLOCK_GCLK_5_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_5_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_5_PRESCALER             1
#  define CONF_CLOCK_GCLK_5_OUTPUT_ENABLE         false

/* Configure GCLK generator 6 */
#  define CONF_CLOCK_GCLK_6_ENABLE                false
#  define CONF_CLOCK_GCLK_6_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_6_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_6_PRESCALER             1
#  define CONF_CLOCK_GCLK_6_OUTPUT_ENABLE         false

/* Configure GCLK generator 7 */
#  define CONF_CLOCK_GCLK_7_ENABLE                false
#  define CONF_CLOCK_GCLK_7_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_7_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_7_PRESCALER             1
#  define CONF_CLOCK_GCLK_7_OUTPUT_ENABLE         false

#endif /* CONF_CLOCKS_H_INCLUDED */
```

10.3.4 conf_board.h

10.3.4.1 AT32UC3A0, AT32UC3A1, AT32UC3B Devices (USBB)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

10.3.4.2 AT32UC3A3, AT32UC3A4 Devices (USBB with High Speed Support)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

10.3.4.3 AT32UC3C, ATUCXXD, ATUCXXL3U, ATUCXXL4U Devices (USBC)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

10.3.4.4 SAM3X, SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// USB pins are used
#define CONF_BOARD_USB_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

10.3.4.5 SAMD21 Devices (USB)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

/* Enable USB VBUS detect */
#define CONF_BOARD_USB_VBUS_DETECT

#endif /* CONF_BOARD_H_INCLUDED */
```

11. USB Host Interface (UHI) for Mass Storage Class (MSC)

USB Host Interface (UHI) for Mass Storage Class (MSC) provides an interface for the configuration and management of USB MSC host.

The outline of this documentation is as follows:

- API Overview
- Quick Start Guide for USB Host Mass-Storage Module (UHI MSC)
- Configuration File Examples

For more details for Atmel® Software Framework (ASF) USB Host Stack, refer to following application note:

- AVR4950: ASF - USB Host Stack[1]

11.1 API Overview

11.1.1 Variable and Type Definitions

11.1.1.1 Type uhi_msc_scsi_callback_t

```
typedef void(* uhi_msc_scsi_callback_t )(bool)
```

Callback type used by uhi_msc_scsi() functions.

11.1.2 Structure Definitions

11.1.2.1 Struct uhi_msc_lun_t

LUN structure information.

Table 11-1. Members

Type	Name	Description
bool	b_write_protected	
struct sbc_read_capacity10_data	capacity	
lun_status_t	status	

11.1.3 Macro Definitions

11.1.3.1 Interface with USB Host Core (UHC)

Definition and functions required by UHC.

Macro UHI_MSC

```
#define UHI_MSC \
  { \
 .install = uhi_msc_install, \
```

[1] http://www.atmel.com/dyn/resources/prod_documents/doc8486.pdf

```
  .enable = uhi_msc_enable, \
  .uninstall = uhi_msc_uninstall, \
  .sof_notify = NULL, \
}
```

Global definition which contains standard UHI API for UHC. It must be added in USB_HOST_UHI definition from conf_usb_host.h file.

11.1.4 Function Definitions

11.1.4.1 Functions Required by UHC

Function uhi_msc_install()

Install interface.

```
uhc_enum_status_t uhi_msc_install(
  uhc_device_t * dev)
```

Allocate interface endpoints if supported.

Table 11-2. Parameters

Data direction	Parameter name	Description
[in]	uhc_device_t	Device to request

Returns Status of the install.

Function uhi_msc_enable()

Enable the interface.

```
void uhi_msc_enable(
  uhc_device_t * dev)
```

Enable a USB interface corresponding to UHI.

Table 11-3. Parameters

Data direction	Parameter name	Description
[in]	uhc_device_t	Device to request

Function uhi_msc_uninstall()

Uninstall the interface (if installed).

```
void uhi_msc_uninstall(
  uhc_device_t * dev)
```

Table 11-4. Parameters

Data direction	Parameter name	Description
[in]	uhc_device_t	Device to request

11.1.4.2 UHI for Mass Storage Class

Common APIs used by high level application to use this USB host class.

Function uhi_msc_is_available()

Tests if the interface UHI Mass Storage is available.

```
bool uhi_msc_is_available(void)
```

The UHI Mass Storage can be busy during the enumeration of a USB Device MSC.

Returns True, if UHI Mass Storage is available.

Function uhi_msc_get_lun()

Gives the number of LUN available.

```
uint8_t uhi_msc_get_lun(void)
```

Note A LUN can be available, but with a status LUN_NOT_PRESENT.

It is the case for a card reader without card.

Returns Number of LUN available.

Function uhi_msc_get_lun_desc()

Gives information about a LUN.

```
uhi_msc_lun_t * uhi_msc_get_lun_desc(
  uint8_t lun)
```

Table 11-5. Parameters

Data direction	Parameter name	Description
[in]	lun	LUN number

Returns Pointer on the LUN information structure.

Function uhi_msc_scsi_test_unit_ready()

Checks and update the status of the LUN.

```
bool uhi_msc_scsi_test_unit_ready(
  uint8_t lun,
  uhi_msc_scsi_callback_t callback)
```

Table 11-6. Parameters

Data direction	Parameter name	Description
[in]	lun	LUN number
[in]	callback	Callback to call at the end of SCSI command

Returns True, if the SCSI command has been accepted.

Function uhi_msc_scsi_read_10()

Reads a LUN data section to RAM buffer.

```
bool uhi_msc_scsi_read_10(
  uint8_t lun,
  uint32_t addr,
  uint8_t * ram,
  uint8_t nb_sector,
  uhi_msc_scsi_callback_t callback)
```

Note The sector size used to define the data section is the sector size returned by LUN in field.

Table 11-7. Parameters

Data direction	Parameter name	Description
[in]	lun	LUN number
[in]	addr	Sector address to read
[out]	ram	RAM address used to store the data
[in]	nb_sector	Number of sector to read
[in]	callback	Callback to call at the end of SCSI command

Returns True, if the SCSI command has been accepted.

Function uhi_msc_scsi_write_10()

Writes a RAM buffer in a LUN data section.

```
bool uhi_msc_scsi_write_10(
  uint8_t lun,
  uint32_t addr,
  const uint8_t * ram,
  uint8_t nb_sector,
  uhi_msc_scsi_callback_t callback)
```

Note The sector size used to define the data section is the sector size returned by LUN in field.

Table 11-8. Parameters

Data direction	Parameter name	Description
[in]	lun	LUN number
[in]	addr	Sector address to write
[in]	ram	RAM address of data to write
[in]	nb_sector	Number of sector to write
[in]	callback	Callback to call at the end of SCSI command

Returns True, if the SCSI command has been accepted.

11.1.4.3 USB Host Mass Storage Interface for Control Access Module

Layer added on UHI MSC interface to allow the usage of control access module. The control access module provides a common access at all memories and it is used by the File Systems available in ASF.

Function uhi_msc_mem_get_lun()

Gives the number of available LUN.

```
uint8_t uhi_msc_mem_get_lun(void)
```

Note A LUN can be available, but with a status not present.

It is the case for a card reader without card.

Returns Number of available LUN.

Function uhi_msc_mem_test_unit_ready()

Checks and update the status of the LUN.

```
Ctrl_status uhi_msc_mem_test_unit_ready(
```

```
  uint8_t lun)
```

Table 11-9. Parameters

Data direction	Parameter name	Description
[in]	lun	LUN number

Returns Status of the LUN.

Function uhi_msc_mem_read_capacity()

Returns the capacity of the LUN.

```
Ctrl_status uhi_msc_mem_read_capacity(
  uint8_t lun,
  uint32_t * u32_nb_sector)
```

Table 11-10. Parameters

Data direction	Parameter name	Description
[in]	lun	LUN number
[in]	u32_nb_sector	Pointer to store the last sector address possible on this LUN

Returns Status of the LUN.

Function uhi_msc_mem_read_sector_size()

Returns the sector size of the LUN.

```
uint8_t uhi_msc_mem_read_sector_size(
  uint8_t lun)
```

Table 11-11. Parameters

Data direction	Parameter name	Description
[in]	lun	LUN number

Returns Sector size (unit 512B).

Function uhi_msc_mem_wr_protect()

Checks if the LUN is write protected.

```
bool uhi_msc_mem_wr_protect(
```

```
  uint8_t lun)
```

Table 11-12. Parameters

Data direction	Parameter name	Description
[in]	lun	LUN number

Returns True, if write protected.

Function uhi_msc_mem_removal()

Checks if the device is removed.

```
bool uhi_msc_mem_removal(void)
```

Returns Always true for USB Device.

Function uhi_msc_mem_read_10_ram()

Reads 512 bytes from the current LUN.

```
Ctrl_status uhi_msc_mem_read_10_ram(
  uint32_t addr,
  void * ram)
```

The LUN is selected by uhi_msc_mem_test_unit_ready() or uhi_msc_mem_read_capacity() function.

Table 11-13. Parameters

Data direction	Parameter name	Description
[in]	addr	Disk address (unit 512B)
[out]	ram	Pointer to store the data

Returns Status of the LUN.

Function uhi_msc_mem_write_10_ram()

Writes 512 bytes to the current LUN.

```
Ctrl_status uhi_msc_mem_write_10_ram(
  uint32_t addr,
  const void * ram)
```

The LUN is selected by uhi_msc_mem_test_unit_ready() or uhi_msc_mem_read_capacity() function.

Table 11-14. Parameters

Data direction	Parameter name	Description
[in]	addr	Disk address (unit 512B)
[in]	ram	Pointer on the data

Returns Status of the LUN.

11.1.5 Enumeration Definitions

11.1.5.1 Enum lun_status_t

Status of LUN.

Table 11-15. Members

Enum value	Description
LUN_GOOD	Success, memory ready
LUN_FAIL	An error occurred
LUN_NOT_PRESENT	Memory unplugged
LUN_BUSY	Memory not initialized or changed

11.2 Quick Start Guide for USB Host Mass-Storage Module (UHI MSC)

This is the quick start guide for the USB Host Mass-Storage Module (UHI MSC) with step-by-step instructions on how to configure and use the modules in a selection of use cases.

The use cases highlights several code fragments. The code fragments in the steps for setup can be copied into a custom initialization function, while the steps for usage can be copied into, e.g., the main application function.

11.2.1 Basic Use Case

In this basic use case, the "USB Host MSC (Single Class support)" module is used.

The "USB Host MSC (Multiple Classes support)" module usage is described in Advanced Use Cases.

This example do a simple physical memory access, but a file system module can be added to decode the USB memory file system, see FatFS examples.

11.2.1.1 Setup Steps

As a USB host, it follows common USB host setup steps. Refer to USB Host Basic Setup.

11.2.1.2 Usage Steps

Example Code

Content of conf_usb_host.h:

```
#define USB_HOST_UHI        UHI_MSC
#define UHI_MSC_CHANGE(dev, b_plug) my_callback_msc_change(dev, b_plug)
extern bool my_callback_msc_change(uhc_device_t* dev, bool b_plug);
#include "uhi_msc_mem.h" // At the end of conf_usb_host.h file
```

Add to application C-file:

```
static bool my_flag_autorize_msc_check = false;
bool my_callback_msc_change(uhc_device_t* dev, bool b_plug)
{
   if (b_plug) {
      my_flag_autorize_msc_check = true;
   } else {
      my_flag_autorize_msc_check = false;
   }
}

void my_task(void)
{
   if (!my_flag_autorize_msc_check) {
      return;
   }
   my_flag_autorize_msc_check = false;

   // Check all new USB disks plugged
   for (uint8_t lun=0; lun < uhi_msc_mem_get_lun(); lun++) {
      // Wait the end of USB disk install
      while (CTRL_BUSY == uhi_msc_mem_test_unit_ready(lun));
      if (CTRL_GOOD != uhi_msc_mem_test_unit_ready(lun)) {
         // Removal disk not present or fail
         continue;
      }
      // Read capacity
      uint32_t max_lba;
      uhi_msc_mem_read_capacity(lun, &max_lba);
   }
}
```

Workflow

1. Ensure that conf_usb_host.h is available and contains the following configuration, which is the USB host MSC configuration:

```
#define USB_HOST_UHI   UHI_MSC
```

Note It defines the list of UHI supported by USB host.

```
#define UHI_MSC_CHANGE(dev, b_plug) my_callback_msc_change(dev, b_plug)
extern bool my_callback_msc_change(uhc_device_t* dev, bool b_plug);
```

Note This callback is called when a USB device MSC is plugged or unplugged.

2. The access of the USB memories is allowed through functions described in API Overview.

11.2.2 Advanced Use Cases

For more advanced use of the UHI MSC module, see the following:

- USB Host Advanced Use Cases

11.3 Configuration File Examples

11.3.1 conf_usb_host.h

11.3.1.1 UHI MSC Single

```
#ifndef _CONF_USB_HOST_H_
#define _CONF_USB_HOST_H_

#include "compiler.h"

#define USB_HOST_UHI        UHI_CDC

#define USB_HOST_POWER_MAX  500

// #define USB_HOST_HUB_SUPPORT

#if (UC3A3||UC3A4)
#  define USB_HOST_HS_SUPPORT
#endif

//#define UHC_MODE_CHANGE(b_host_mode)        usb_host_mode_change(b_host_mode)

//#define UHC_VBUS_CHANGE(b_present)          usb_host_vbus_change(b_present)

//#define UHC_VBUS_ERROR()                    usb_host_vbus_error()

//#define UHC_CONNECTION_EVENT(dev,b_present) usb_host_connection_event(dev,b_present)

//#define UHC_WAKEUP_EVENT()                  usb_host_wakeup_event()

//#define UHC_SOF_EVENT()                     usb_host_sof_event()

//#define UHC_DEVICE_CONF(dev)                uint8_t usb_host_device_conf(dev)

//#define UHC_ENUM_EVENT(dev,b_status)        usb_host_enum_event(dev,b_status)




#define UHI_CDC_CHANGE(dev,b_plug)
#define UHI_CDC_RX_NOTIFY()



#include "uhi_cdc.h"

#endif // _CONF_USB_HOST_H_
```

11.3.1.2 UHI MSC Multiple (Composite)

```
#ifndef _CONF_USB_HOST_H_
#define _CONF_USB_HOST_H_

#include "compiler.h"

#define USB_HOST_UHI        // UHI_MSC, UHI_HID_MOUSE, UHI_CDC, UHI_VENDOR
```

```
#define USB_HOST_POWER_MAX  500

// #define USB_HOST_HUB_SUPPORT

#if (UC3A3 || UC3A4)
#  define USB_HOST_HS_SUPPORT
#endif

//#define UHC_MODE_CHANGE(b_host_mode)        usb_host_mode_change(b_host_mode)

//#define UHC_VBUS_CHANGE(b_present)          usb_host_vbus_change(b_present)

//#define UHC_VBUS_ERROR()                    usb_host_vbus_error()

//#define UHC_CONNECTION_EVENT(dev,b_present) usb_host_connection_event(dev,b_present)

//#define UHC_WAKEUP_EVENT()                  usb_host_wakeup_event()

//#define UHC_SOF_EVENT()                     usb_host_sof_event()

//#define UHC_DEVICE_CONF(dev)                uint8_t usb_host_device_conf(dev)

//#define UHC_ENUM_EVENT(dev,b_status)        usb_host_enum_event(dev,b_status)




#define UHI_HID_MOUSE_CHANGE(dev,b_plug)
#define UHI_HID_MOUSE_EVENT_BTN_LEFT(b_state)
#define UHI_HID_MOUSE_EVENT_BTN_RIGHT(b_state)
#define UHI_HID_MOUSE_EVENT_BTN_MIDDLE(b_state)
#define UHI_HID_MOUSE_EVENT_MOUVE(x,y,scroll)


#define UHI_MSC_CHANGE(dev,b_plug)


#define UHI_CDC_CHANGE(dev,b_plug)
#define UHI_CDC_RX_NOTIFY()


#define UHI_VENDOR_CHANGE(dev, b_plug)
#define UHI_VENDOR_VID_PID_LIST {USB_VID_ATMEL, USB_PID_ATMEL_ASF_VENDOR_CLASS}




//#include "uhi_msc.h"
//#include "uhi_hid_mouse.h"

#endif // _CONF_USB_HOST_H_
```

11.3.2 conf_clock.h

11.3.2.1 AT32UC3A0, AT32UC3A1, and AT32UC3B Devices (USBB)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED
```

```
// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL0

// ===== PLL0 Options
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_MUL             4 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL0_DIV             1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
#define CONFIG_PLL1_SOURCE            PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
#define CONFIG_PLL1_MUL               8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL1_DIV               2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
//#define CONFIG_SYSCLK_CPU_DIV         0 /* Fcpu = Fsys/(2 ^ CPU_div) */
//#define CONFIG_SYSCLK_PBA_DIV         0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV         0 /* Fpbb = Fsys/(2 ^ PBB_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

11.3.2.2 AT32UC3A3 and AT32UC3A4 Devices (USBB with High Speed Support)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL0

// ===== PLL0 Options
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_MUL             11 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL0_DIV             2  /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL1_MUL             8 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL1_DIV             2 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
```

```
#define CONFIG_SYSCLK_CPU_DIV         0 /* Fcpu = Fsys/(2 ^ CPU_div) */
#define CONFIG_SYSCLK_PBA_DIV         0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV       0 /* Fpbb = Fsys/(2 ^ PBB_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

11.3.2.3 AT32UC3C, ATUCXXD, ATUCXXL3U, and ATUCXXL4U Devices (USBC)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC1
#define CONFIG_SYSCLK_SOURCE          SYSCLK_SRC_PLL0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL1
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RC8M

// ===== PLL0 Options
#define CONFIG_PLL0_SOURCE            PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL0_SOURCE          PLL_SRC_RC8M
#define CONFIG_PLL0_MUL               3 /* Fpll = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL0_DIV               1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== PLL1 Options
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL1_SOURCE          PLL_SRC_OSC1
//#define CONFIG_PLL1_SOURCE          PLL_SRC_RC8M
//#define CONFIG_PLL1_MUL             3 /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL1_DIV             1 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ===== System Clock Bus Division Options
//#define CONFIG_SYSCLK_CPU_DIV       0 /* Fcpu = Fsys/(2 ^ CPU_div) */
//#define CONFIG_SYSCLK_PBA_DIV       0 /* Fpba = Fsys/(2 ^ PBA_div) */
//#define CONFIG_SYSCLK_PBB_DIV       0 /* Fpbb = Fsys/(2 ^ PBB_div) */
//#define CONFIG_SYSCLK_PBC_DIV       0 /* Fpbc = Fsys/(2 ^ PBC_div) */

// ===== Peripheral Clock Management Options
//#define CONFIG_SYSCLK_INIT_CPUMASK  ((1 << SYSCLK_SYSTIMER) | (1 << SYSCLK_OCD))
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_USART0)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_HMATRIX)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_MDMA_HSB)

// ===== USB Clock Source Options
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_OSC1
```

```
#define CONFIG_USBCLK_SOURCE          USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL1
#define CONFIG_USBCLK_DIV             1 /* Fusb = Fsys/(2 ^ USB_div) */

#endif /* CONF_CLOCK_H_INCLUDED */
```

11.3.2.4 SAM3X and SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock (MCK) Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_BYPASS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_4M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_8M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_12M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_BYPASS
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLLACK
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_UPLLCK

// ===== System Clock (MCK) Prescaler Options   (Fmck = Fsys / (SYSCLK_PRES))
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_1
#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_2
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_4
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_8
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_16
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_32
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_64
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_3

// ===== PLL0 (A) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
// Use mul and div effective values here.
#define CONFIG_PLL0_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL0_MUL             14
#define CONFIG_PLL0_DIV             1

// ===== UPLL (UTMI) Hardware fixed at 480MHz.

// ===== USB Clock Source Options   (Fusb = FpllX / USB_div)
// Use div effective value here.
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_UPLL
#define CONFIG_USBCLK_DIV           1

// ===== Target frequency (System clock)
// - XTAL frequency: 12MHz
// - System clock source: PLLA
// - System clock prescaler: 2 (divided by 2)
// - PLLA source: XTAL
// - PLLA output: XTAL * 14 / 1
// - System clock is: 12 * 14 / 1 /2 = 84MHz
// ===== Target frequency (USB Clock)
// - USB clock source: UPLL
// - USB clock divider: 1 (not divided)
// - UPLL frequency: 480MHz
// - USB clock: 480 / 1 = 480MHz
```

```
#endif /* CONF_CLOCK_H_INCLUDED */
```

11.3.3 conf_clocks.h

11.3.3.1 SAMD21 Devices (USB)

```
#include <clock.h>

#ifndef CONF_CLOCKS_H_INCLUDED
#  define CONF_CLOCKS_H_INCLUDED

/* System clock bus configuration */
#  define CONF_CLOCK_CPU_CLOCK_FAILURE_DETECT     false
#  define CONF_CLOCK_FLASH_WAIT_STATES            2
#  define CONF_CLOCK_CPU_DIVIDER                  SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBA_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBB_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1

/* SYSTEM_CLOCK_SOURCE_OSC8M configuration - Internal 8MHz oscillator */
#  define CONF_CLOCK_OSC8M_PRESCALER              SYSTEM_OSC8M_DIV_1
#  define CONF_CLOCK_OSC8M_ON_DEMAND              true
#  define CONF_CLOCK_OSC8M_RUN_IN_STANDBY         true

/* SYSTEM_CLOCK_SOURCE_XOSC configuration - External clock/oscillator */
#  define CONF_CLOCK_XOSC_ENABLE                  false
#  define CONF_CLOCK_XOSC_EXTERNAL_CRYSTAL        SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC_EXTERNAL_FREQUENCY      12000000UL
#  define CONF_CLOCK_XOSC_STARTUP_TIME            SYSTEM_XOSC_STARTUP_32768
#  define CONF_CLOCK_XOSC_AUTO_GAIN_CONTROL       true
#  define CONF_CLOCK_XOSC_ON_DEMAND               true
#  define CONF_CLOCK_XOSC_RUN_IN_STANDBY          false

/* SYSTEM_CLOCK_SOURCE_XOSC32K configuration - External 32KHz crystal/clock oscillator */
#  define CONF_CLOCK_XOSC32K_ENABLE               true
#  define CONF_CLOCK_XOSC32K_EXTERNAL_CRYSTAL     SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC32K_STARTUP_TIME         SYSTEM_XOSC32K_STARTUP_65536
#  define CONF_CLOCK_XOSC32K_AUTO_AMPLITUDE_CONTROL  false
#  define CONF_CLOCK_XOSC32K_ENABLE_1KHZ_OUPUT    false
#  define CONF_CLOCK_XOSC32K_ENABLE_32KHZ_OUTPUT  true
#  define CONF_CLOCK_XOSC32K_ON_DEMAND            false
#  define CONF_CLOCK_XOSC32K_RUN_IN_STANDBY       true

/* SYSTEM_CLOCK_SOURCE_OSC32K configuration - Internal 32KHz oscillator */
#  define CONF_CLOCK_OSC32K_ENABLE                false
#  define CONF_CLOCK_OSC32K_STARTUP_TIME          SYSTEM_OSC32K_STARTUP_130
#  define CONF_CLOCK_OSC32K_ENABLE_1KHZ_OUTPUT    false
#  define CONF_CLOCK_OSC32K_ENABLE_32KHZ_OUTPUT   true
#  define CONF_CLOCK_OSC32K_ON_DEMAND             true
#  define CONF_CLOCK_OSC32K_RUN_IN_STANDBY        false

/* SYSTEM_CLOCK_SOURCE_DFLL configuration - Digital Frequency Locked Loop */
#  define CONF_CLOCK_DFLL_ENABLE                  true
#  define CONF_CLOCK_DFLL_LOOP_MODE               SYSTEM_CLOCK_DFLL_LOOP_MODE_CLOSED
#  define CONF_CLOCK_DFLL_ON_DEMAND               true

/* DFLL open loop mode configuration */
#  define CONF_CLOCK_DFLL_COARSE_VALUE            (0x1f / 4)
#  define CONF_CLOCK_DFLL_FINE_VALUE              (0xff / 4)
```

```
/* DFLL closed loop mode configuration */
#  define CONF_CLOCK_DFLL_SOURCE_GCLK_GENERATOR   GCLK_GENERATOR_1
#  define CONF_CLOCK_DFLL_MULTIPLY_FACTOR         (48000000/32768)
#  define CONF_CLOCK_DFLL_QUICK_LOCK              true
#  define CONF_CLOCK_DFLL_TRACK_AFTER_FINE_LOCK   true
#  define CONF_CLOCK_DFLL_KEEP_LOCK_ON_WAKEUP     true
#  define CONF_CLOCK_DFLL_ENABLE_CHILL_CYCLE      true
#  define CONF_CLOCK_DFLL_MAX_COARSE_STEP_SIZE    (0x1f / 8)
#  define CONF_CLOCK_DFLL_MAX_FINE_STEP_SIZE      (0xff / 8)

/* SYSTEM_CLOCK_SOURCE_DPLL configuration - Digital Phase-Locked Loop */
#  define CONF_CLOCK_DPLL_ENABLE                  false
#  define CONF_CLOCK_DPLL_ON_DEMAND               false
#  define CONF_CLOCK_DPLL_RUN_IN_STANDBY          true
#  define CONF_CLOCK_DPLL_LOCK_BYPASS             false
#  define CONF_CLOCK_DPLL_WAKE_UP_FAST            false
#  define CONF_CLOCK_DPLL_LOW_POWER_ENABLE        true

#  define CONF_CLOCK_DPLL_LOCK_TIME               SYSTEM_CLOCK_SOURCE_DPLL_LOCK_TIME_NO_TIMEOUT
#  define CONF_CLOCK_DPLL_REFERENCE_CLOCK         SYSTEM_CLOCK_SOURCE_DPLL_REFERENCE_CLOCK_REF0
#  define CONF_CLOCK_DPLL_FILTER                  SYSTEM_CLOCK_SOURCE_DPLL_FILTER_DEFAULT

#  define CONF_CLOCK_DPLL_REFERENCE_FREQUENCY     32768
#  define CONF_CLOCK_DPLL_REFEREMCE_DIVIDER       1
#  define CONF_CLOCK_DPLL_OUTPUT_FREQUENCY        48000000

/* Set this to true to configure the GCLK when running clocks_init. If set to
 * false, none of the GCLK generators will be configured in clocks_init(). */
#  define CONF_CLOCK_CONFIGURE_GCLK               true

/* Configure GCLK generator 0 (Main Clock) */
#  define CONF_CLOCK_GCLK_0_ENABLE                true
#  define CONF_CLOCK_GCLK_0_RUN_IN_STANDBY        true
#  define CONF_CLOCK_GCLK_0_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_DFLL
#  define CONF_CLOCK_GCLK_0_PRESCALER             1
#  define CONF_CLOCK_GCLK_0_OUTPUT_ENABLE         false

/* Configure GCLK generator 1 */
#  define CONF_CLOCK_GCLK_1_ENABLE                true
#  define CONF_CLOCK_GCLK_1_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_1_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_XOSC32K
#  define CONF_CLOCK_GCLK_1_PRESCALER             1
#  define CONF_CLOCK_GCLK_1_OUTPUT_ENABLE         false

/* Configure GCLK generator 2 (RTC) */
#  define CONF_CLOCK_GCLK_2_ENABLE                false
#  define CONF_CLOCK_GCLK_2_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_2_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC32K
#  define CONF_CLOCK_GCLK_2_PRESCALER             32
#  define CONF_CLOCK_GCLK_2_OUTPUT_ENABLE         false

/* Configure GCLK generator 3 */
#  define CONF_CLOCK_GCLK_3_ENABLE                false
#  define CONF_CLOCK_GCLK_3_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_3_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_3_PRESCALER             1
#  define CONF_CLOCK_GCLK_3_OUTPUT_ENABLE         false

/* Configure GCLK generator 4 */
#  define CONF_CLOCK_GCLK_4_ENABLE                false
#  define CONF_CLOCK_GCLK_4_RUN_IN_STANDBY        false
```

```
#  define CONF_CLOCK_GCLK_4_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_4_PRESCALER             1
#  define CONF_CLOCK_GCLK_4_OUTPUT_ENABLE         false

/* Configure GCLK generator 5 */
#  define CONF_CLOCK_GCLK_5_ENABLE                false
#  define CONF_CLOCK_GCLK_5_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_5_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_5_PRESCALER             1
#  define CONF_CLOCK_GCLK_5_OUTPUT_ENABLE         false

/* Configure GCLK generator 6 */
#  define CONF_CLOCK_GCLK_6_ENABLE                false
#  define CONF_CLOCK_GCLK_6_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_6_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_6_PRESCALER             1
#  define CONF_CLOCK_GCLK_6_OUTPUT_ENABLE         false

/* Configure GCLK generator 7 */
#  define CONF_CLOCK_GCLK_7_ENABLE                false
#  define CONF_CLOCK_GCLK_7_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_7_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_7_PRESCALER             1
#  define CONF_CLOCK_GCLK_7_OUTPUT_ENABLE         false

#endif /* CONF_CLOCKS_H_INCLUDED */
```

11.3.4 conf_board.h

11.3.4.1 AT32UC3A0, AT32UC3A1, and AT32UC3B Devices (USBB)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

11.3.4.2 AT32UC3A3, and AT32UC3A4 Devices (USBB with High Speed Support)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

11.3.4.3 AT32UC3C, ATUCXXD, ATUCXXL3U, and ATUCXXL4U Devices (USBC)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Only the default board init (switchs/leds) is necessary for this example

#endif /* CONF_BOARD_H_INCLUDED */
```

11.3.4.4 SAM3X and SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// UART module is used
#define CONF_BOARD_UART_CONSOLE
// USART0 module is used
#define CONF_BOARD_USART_RXD
#define CONF_BOARD_USART_TXD
#define CONF_BOARD_ADM3312_EN
// USB pins are used
#define CONF_BOARD_USB_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

11.3.4.5 SAMD21 Devices (USB)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

/* Enable USB VBUS detect */
#define CONF_BOARD_USB_VBUS_DETECT
/* ID detect enabled */
#define CONF_BOARD_USB_ID_DETECT

#endif /* CONF_BOARD_H_INCLUDED */
```

12. USB Host Interface (UHI) for Vendor Class Device

USB Host Interface (UHI) for Vendor Class Device provides an interface for the configuration and management of USB Vendor host.

The outline of this documentation is as follows:

- API Overview
- Quick Start Guide for USB Host Vendor Module (UHI Vendor)
- Configuration File Examples

For more details for Atmel® Software Framework (ASF) USB Host Stack, refer to following application note:

- AVR4950: ASF - USB Host Stack[1]

12.1 API Overview

12.1.1 Macro Definitions

12.1.1.1 Interface with USB Host Core (UHC)

Definition and functions required by UHC.

Macro UHI_VENDOR

```
#define UHI_VENDOR \
  { \
 .install = uhi_vendor_install, \
 .enable = uhi_vendor_enable, \
 .uninstall = uhi_vendor_uninstall, \
 .sof_notify = NULL, \
}
```

Global definition which contains standard UHI API for UHC It must be added in USB_HOST_UHI definition from conf_usb_host.h file.

12.1.2 Function Definitions

12.1.2.1 Functions Required by UHC

Function uhi_vendor_install()

Install interface.

```
uhc_enum_status_t uhi_vendor_install(
  uhc_device_t * dev)
```

Allocate interface endpoints if supported.

Table 12-1. Parameters

Data direction	Parameter name	Description
[in]	uhc_device_t	Device to request

[1] http://www.atmel.com/dyn/resources/prod_documents/doc8486.pdf

Returns Status of the install.

Function uhi_vendor_enable()

Enable the interface.

```
void uhi_vendor_enable(
  uhc_device_t * dev)
```

Enable a USB interface corresponding to UHI.

Table 12-2. Parameters

Data direction	Parameter name	Description
[in]	uhc_device_t	Device to request

Function uhi_vendor_uninstall()

Uninstall the interface (if installed).

```
void uhi_vendor_uninstall(
  uhc_device_t * dev)
```

Table 12-3. Parameters

Data direction	Parameter name	Description
[in]	uhc_device_t	Device to request

12.1.2.2 UHI for Vendor Class

Common APIs used by high level application to use this USB host class.

This Vendor Class implementation supports one endpoint for all endpoint types on all directions: Control IN, control OUT, interrupt IN, interrupt OUT, bulk IN, bulk OUT, isochronous IN, isochronous OUT.

This implementation is an example and can be a base to create another Vendor Class which supports more endpoint as two bulk IN endpoints.

Function uhi_vendor_control_in_run()

Start a transfer on control IN.

```
bool uhi_vendor_control_in_run(
  uint8_t * buf,
  iram_size_t buf_size,
  uhd_callback_setup_end_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 12-4. Parameters

Data direction	Parameter name	Description
[out]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.

Data direction	Parameter name	Description
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns 1 if function was successfully done, otherwise 0.

Function uhi_vendor_control_out_run()

Start a transfer on control OUT.

```
bool uhi_vendor_control_out_run(
  uint8_t * buf,
  iram_size_t buf_size,
  uhd_callback_setup_end_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 12-5. Parameters

Data direction	Parameter name	Description
[in]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns 1 if function was successfully done, otherwise 0.

Function uhi_vendor_bulk_in_run()

Start a transfer on bulk IN.

```
bool uhi_vendor_bulk_in_run(
  uint8_t * buf,
  iram_size_t buf_size,
  uhd_callback_trans_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 12-6. Parameters

Data direction	Parameter name	Description
[out]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.

Data direction	Parameter name	Description
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns 1 if function was successfully done, otherwise 0.

Function uhi_vendor_bulk_out_run()

Start a transfer on bulk OUT.

```
bool uhi_vendor_bulk_out_run(
  uint8_t * buf,
  iram_size_t buf_size,
  uhd_callback_trans_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 12-7. Parameters

Data direction	Parameter name	Description
[in]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns 1 if function was successfully done, otherwise 0.

Function uhi_vendor_int_in_run()

Start a transfer on interrupt IN.

```
bool uhi_vendor_int_in_run(
  uint8_t * buf,
  iram_size_t buf_size,
  uhd_callback_trans_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 12-8. Parameters

Data direction	Parameter name	Description
[out]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.

Data direction	Parameter name	Description
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns 1 if function was successfully done, otherwise 0.

Function uhi_vendor_int_out_run()

Start a transfer on interrupt OUT.

```
bool uhi_vendor_int_out_run(
  uint8_t * buf,
  iram_size_t buf_size,
  uhd_callback_trans_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 12-9. Parameters

Data direction	Parameter name	Description
[in]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns 1 if function was successfully done, otherwise 0.

Function uhi_vendor_iso_in_run()

Start a transfer on ISO IN.

```
bool uhi_vendor_iso_in_run(
  uint8_t * buf,
  iram_size_t buf_size,
  uhd_callback_trans_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 12-10. Parameters

Data direction	Parameter name	Description
[out]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.

Data direction	Parameter name	Description
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns 1 if function was successfully done, otherwise 0.

Function uhi_vendor_iso_out_run()

Start a transfer on ISO OUT.

```
bool uhi_vendor_iso_out_run(
  uint8_t * buf,
  iram_size_t buf_size,
  uhd_callback_trans_t callback)
```

When the transfer is finished or aborted (stall, reset, ...), the *callback* is called. The *callback* returns the transfer status and eventually the number of byte transferred.

Table 12-11. Parameters

Data direction	Parameter name	Description
[in]	buf	Buffer on Internal RAM to send or fill. It must be align, then use COMPILER_WORD_ALIGNED.
[in]	buf_size	Buffer size to send or fill
[in]	callback	NULL or function to call at the end of transfer

Returns 1 if function was successfully done, otherwise 0.

Function uhi_vendor_bulk_is_available()

Check if a transfer on BULK is possible.

```
bool uhi_vendor_bulk_is_available(void)
```

Returns 1 if possible, otherwise 0.

Function uhi_vendor_int_is_available()

Check if a transfer on INTERRUPT is possible.

```
bool uhi_vendor_int_is_available(void)
```

Returns 1 if possible, otherwise 0.

Function uhi_vendor_iso_is_available()

Check if a transfer on ISO is possible.

```
bool uhi_vendor_iso_is_available(void)
```

Returns 1 if possible, otherwise 0.

12.2 Quick Start Guide for USB Host Vendor Module (UHI Vendor)

This is the quick start guide for the USB Host Vendor Module (UHI Vendor) with step-by-step instructions on how to configure and use the modules in a selection of use cases.

The use cases highlights several code fragments. The code fragments in the steps for setup can be copied into a custom initialization function, while the steps for usage can be copied into, e.g., the main application function.

12.2.1 Basic Use Case

In this basic use case, the "USB Vendor (Single Class support)" module is used.

The "USB Vendor (Composite)" module usage is described in Advanced Use Cases.

12.2.1.1 Setup Steps

As a USB host, it follows common USB host setup steps. Refer to USB Host Basic Setup.

12.2.1.2 Usage Steps

Example Code

Content of conf_usb_host.h:

```
#define USB_HOST_UHI        UHI_VENDOR
#define UHI_VENDOR_CHANGE(dev, b_plug) my_callback_vendor_change(dev, b_plug)
extern void my_callback_vendor_change(uhc_device_t* dev, bool b_plug);
#define UHI_VENDOR_VID_PID_LIST {USB_VID_ATMEL, USB_PID_ATMEL_ASF_VENDOR_CLASS}
#include "uhi_vendor.h" // At the end of conf_usb_host.h file
```

Add to application C-file:

```
static bool my_flag_vendor_test_start = false;
void my_callback_vendor_change(uhc_device_t* dev, bool b_plug)
{
   // USB Device Vendor connected
   my_flag_vendor_test_start = b_plug;
}

static void my_callback_bulk_in_done (usb_add_t add,
        usb_ep_t ep, uhd_trans_status_t status, iram_size_t nb_transfered)
```

```
{
  if (status != UHD_TRANS_NOERROR) {
    return; // Error during transfer
  }
  // Data received then restart test
  my_flag_vendor_test_start = true;
}

#define MESSAGE "Hello bulk"
#define HELLO_SIZE 5
#define HELLO_BULK_SIZE 10
uint8_t my_out_buffer[MESSAGE_SIZE+1] = MESSAGE;
uint8_t my_in_buffer[MESSAGE_SIZE+1];
void my_task(void)
{
   if (!my_flag_vendor_test_start) {
     return;
   }
   my_flag_vendor_test_start = false;

   // Send data through control endpoint
   uhi_vendor_control_out_run(my_out_buffer, HELLO_SIZE, NULL);

   // Check if bulk endpoints are available
   if (uhi_vendor_bulk_is_available()) {
     // Send data through bulk OUT endpoint
     uhi_vendor_bulk_out_run(my_out_buffer, HELLO_BULK_SIZE, NULL);
     // Receive data through bulk IN endpoint
     uhi_vendor_bulk_in_run(my_in_buffer, sizeof(my_in_buffer),
             my_callback_bulk_in_done);
   }
}
```

Workflow

1. Ensure that conf_usb_host.h is available and contains the following configurations, which is the USB host vendor configuration:

```
#define USB_HOST_UHI   UHI_HID_VENDOR
```

Note It defines the list of UHI supported by USB host.

```
#define UHI_VENDOR_CHANGE(dev, b_plug) my_callback_vendor_change(dev, b_plug)
extern bool my_callback_vendor_change(uhc_device_t* dev, bool b_plug);
```

Note This callback is called when a USB device vendor is plugged or unplugged.

```
#define UHI_VENDOR_VID_PID_LIST {USB_VID_ATMEL, USB_PID_ATMEL_ASF_VENDOR_CLASS}
```

Note It defines the list of devices supported by USB host (defined by VID and PID).

2. The Vendor data transfert functions are described in uhi_vendor_group.

```
uhi_vendor_control_out_run(), uhi_vendor_bulk_out_run(),...
```

12.2.2 Advanced Use Cases

For more advanced use of the UHI vendor module, see the following:

- USB Host Advanced Use Cases

12.3 Configuration File Examples

12.3.1 conf_usb_host.h

12.3.1.1 UHI Vendor Single

```
#ifndef _CONF_USB_HOST_H_
#define _CONF_USB_HOST_H_

#include "compiler.h"

#define USB_HOST_UHI        UHI_CDC

#define USB_HOST_POWER_MAX  500

// #define USB_HOST_HUB_SUPPORT

#if (UC3A3||UC3A4)
#  define USB_HOST_HS_SUPPORT
#endif

//#define UHC_MODE_CHANGE(b_host_mode)        usb_host_mode_change(b_host_mode)

//#define UHC_VBUS_CHANGE(b_present)          usb_host_vbus_change(b_present)

//#define UHC_VBUS_ERROR()                    usb_host_vbus_error()

//#define UHC_CONNECTION_EVENT(dev,b_present) usb_host_connection_event(dev,b_present)

//#define UHC_WAKEUP_EVENT()                  usb_host_wakeup_event()

//#define UHC_SOF_EVENT()                     usb_host_sof_event()

//#define UHC_DEVICE_CONF(dev)                uint8_t usb_host_device_conf(dev)

//#define UHC_ENUM_EVENT(dev,b_status)        usb_host_enum_event(dev,b_status)




#define UHI_CDC_CHANGE(dev,b_plug)
#define UHI_CDC_RX_NOTIFY()



#include "uhi_cdc.h"

#endif // _CONF_USB_HOST_H_
```

12.3.1.2 UHI Vendor Multiple (Composite)

```
#ifndef _CONF_USB_HOST_H_
#define _CONF_USB_HOST_H_

#include "compiler.h"

#define USB_HOST_UHI        // UHI_MSC, UHI_HID_MOUSE, UHI_CDC, UHI_VENDOR

#define USB_HOST_POWER_MAX  500

// #define USB_HOST_HUB_SUPPORT

#if (UC3A3 || UC3A4)
#  define USB_HOST_HS_SUPPORT
#endif

//#define UHC_MODE_CHANGE(b_host_mode)        usb_host_mode_change(b_host_mode)

//#define UHC_VBUS_CHANGE(b_present)          usb_host_vbus_change(b_present)

//#define UHC_VBUS_ERROR()                    usb_host_vbus_error()

//#define UHC_CONNECTION_EVENT(dev,b_present) usb_host_connection_event(dev,b_present)

//#define UHC_WAKEUP_EVENT()                  usb_host_wakeup_event()

//#define UHC_SOF_EVENT()                     usb_host_sof_event()

//#define UHC_DEVICE_CONF(dev)                uint8_t usb_host_device_conf(dev)

//#define UHC_ENUM_EVENT(dev,b_status)        usb_host_enum_event(dev,b_status)




#define UHI_HID_MOUSE_CHANGE(dev,b_plug)
#define UHI_HID_MOUSE_EVENT_BTN_LEFT(b_state)
#define UHI_HID_MOUSE_EVENT_BTN_RIGHT(b_state)
#define UHI_HID_MOUSE_EVENT_BTN_MIDDLE(b_state)
#define UHI_HID_MOUSE_EVENT_MOUVE(x,y,scroll)


#define UHI_MSC_CHANGE(dev,b_plug)


#define UHI_CDC_CHANGE(dev,b_plug)
#define UHI_CDC_RX_NOTIFY()


#define UHI_VENDOR_CHANGE(dev, b_plug)
#define UHI_VENDOR_VID_PID_LIST {USB_VID_ATMEL, USB_PID_ATMEL_ASF_VENDOR_CLASS}




//#include "uhi_msc.h"
//#include "uhi_hid_mouse.h"
```

```
#endif // _CONF_USB_HOST_H_
```

12.3.2 conf_clock.h

12.3.2.1 SAM3X, and SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

// ===== System Clock (MCK) Source Options
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_SLCK_BYPASS
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_4M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_8M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_12M_RC
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_XTAL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_MAINCK_BYPASS
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLLACK
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_UPLLCK

// ===== System Clock (MCK) Prescaler Options   (Fmck = Fsys / (SYSCLK_PRES))
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_1
#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_2
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_4
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_8
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_16
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_32
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_64
//#define CONFIG_SYSCLK_PRES          SYSCLK_PRES_3

// ===== PLL0 (A) Options   (Fpll = (Fclk * PLL_mul) / PLL_div)
// Use mul and div effective values here.
#define CONFIG_PLL0_SOURCE          PLL_SRC_MAINCK_XTAL
#define CONFIG_PLL0_MUL             14
#define CONFIG_PLL0_DIV             1

// ===== UPLL (UTMI) Hardware fixed at 480MHz.

// ===== USB Clock Source Options   (Fusb = FpllX / USB_div)
// Use div effective value here.
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_UPLL
#define CONFIG_USBCLK_DIV           1

// ===== Target frequency (System clock)
// - XTAL frequency: 12MHz
// - System clock source: PLLA
// - System clock prescaler: 2 (divided by 2)
// - PLLA source: XTAL
// - PLLA output: XTAL * 14 / 1
// - System clock is: 12 * 14 / 1 /2 = 84MHz
// ===== Target frequency (USB Clock)
// - USB clock source: UPLL
// - USB clock divider: 1 (not divided)
// - UPLL frequency: 480MHz
// - USB clock: 480 / 1 = 480MHz


#endif /* CONF_CLOCK_H_INCLUDED */
```

12.3.2.2 SAM4L Device (USBC)

```
#ifndef CONF_CLOCK_H_INCLUDED
#define CONF_CLOCK_H_INCLUDED

//#define CONFIG_SYSCLK_INIT_CPUMASK  (1 << SYSCLK_OCD)
//#define CONFIG_SYSCLK_INIT_PBAMASK  (1 << SYSCLK_IISC)
//#define CONFIG_SYSCLK_INIT_PBBMASK  (1 << SYSCLK_USBC_REGS)
//#define CONFIG_SYSCLK_INIT_PBCMASK  (1 << SYSCLK_CHIPID)
//#define CONFIG_SYSCLK_INIT_PBDMASK  (1 << SYSCLK_AST)
//#define CONFIG_SYSCLK_INIT_HSBMASK  (1 << SYSCLK_PDCA_HSB)

//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCSYS
#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_OSC0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_PLL0
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_DFLL
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RC80M
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RCFAST
//#define CONFIG_SYSCLK_SOURCE        SYSCLK_SRC_RC1M

/* RCFAST frequency selection: 0 for 4MHz, 1 for 8MHz and 2 for 12MHz */
//#define CONFIG_RCFAST_FRANGE    0
//#define CONFIG_RCFAST_FRANGE    1
//#define CONFIG_RCFAST_FRANGE    2

/* Fbus = Fsys / (2 ^ BUS_div) */
#define CONFIG_SYSCLK_CPU_DIV         0
#define CONFIG_SYSCLK_PBA_DIV         0
#define CONFIG_SYSCLK_PBB_DIV         0
#define CONFIG_SYSCLK_PBC_DIV         0
#define CONFIG_SYSCLK_PBD_DIV         0

// ===== Disable all non-essential peripheral clocks
//#define CONFIG_SYSCLK_INIT_CPUMASK  0
//#define CONFIG_SYSCLK_INIT_PBAMASK  SYSCLK_USART1
//#define CONFIG_SYSCLK_INIT_PBBMASK  0
//#define CONFIG_SYSCLK_INIT_PBCMASK  0
//#define CONFIG_SYSCLK_INIT_PBDMASK  0
//#define CONFIG_SYSCLK_INIT_HSBMASK  0

// ===== PLL Options
#define CONFIG_PLL0_SOURCE          PLL_SRC_OSC0
//#define CONFIG_PLL0_SOURCE          PLL_SRC_GCLK9

/* Fpll0 = (Fclk * PLL_mul) / PLL_div */
#define CONFIG_PLL0_MUL             (48000000UL / BOARD_OSC0_HZ)
#define CONFIG_PLL0_DIV             1
//#define CONFIG_PLL0_MUL             (192000000 / FOSC0) /* Fpll = (Fclk * PLL_mul) / PLL_div */
//#define CONFIG_PLL0_DIV             4 /* Fpll = (Fclk * PLL_mul) / PLL_div */

// ==== DFLL Options
//#define CONFIG_DFLL0_SOURCE         GENCLK_SRC_OSC0
//#define CONFIG_DFLL0_SOURCE         GENCLK_SRC_RCSYS
//#define CONFIG_DFLL0_SOURCE         GENCLK_SRC_OSC32K
//#define CONFIG_DFLL0_SOURCE         GENCLK_SRC_RC120M
//#define CONFIG_DFLL0_SOURCE         GENCLK_SRC_RC32K

/* Fdfll = (Fclk * DFLL_mul) / DFLL_div */
//#define CONFIG_DFLL0_FREQ           48000000UL
//#define CONFIG_DFLL0_MUL            ((4 * CONFIG_DFLL0_FREQ) / BOARD_OSC32_HZ)
//#define CONFIG_DFLL0_DIV            4
```

```
//#define CONFIG_DFLL0_MUL            (CONFIG_DFLL0_FREQ / BOARD_OSC32_HZ)
//#define CONFIG_DFLL0_DIV            1

// ===== USB Clock Source Options
#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_PLL0
//#define CONFIG_USBCLK_SOURCE        USBCLK_SRC_DFLL

/* Fusb = Fsys / USB_div */
#define CONFIG_USBCLK_DIV           1

// ===== GCLK9 option
//#define CONFIG_GCLK9_SOURCE           GENCLK_SRC_GCLKIN0
//#define CONFIG_GCLK9_DIV              1

#endif /* CONF_CLOCK_H_INCLUDED */
```

12.3.3 conf_clocks.h

12.3.3.1 SAMD21 Devices (USB)

```
#include <clock.h>

#ifndef CONF_CLOCKS_H_INCLUDED
#  define CONF_CLOCKS_H_INCLUDED

/* System clock bus configuration */
#  define CONF_CLOCK_CPU_CLOCK_FAILURE_DETECT     false
#  define CONF_CLOCK_FLASH_WAIT_STATES            2
#  define CONF_CLOCK_CPU_DIVIDER                  SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBA_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1
#  define CONF_CLOCK_APBB_DIVIDER                 SYSTEM_MAIN_CLOCK_DIV_1

/* SYSTEM_CLOCK_SOURCE_OSC8M configuration - Internal 8MHz oscillator */
#  define CONF_CLOCK_OSC8M_PRESCALER              SYSTEM_OSC8M_DIV_1
#  define CONF_CLOCK_OSC8M_ON_DEMAND              true
#  define CONF_CLOCK_OSC8M_RUN_IN_STANDBY         false

/* SYSTEM_CLOCK_SOURCE_XOSC configuration - External clock/oscillator */
#  define CONF_CLOCK_XOSC_ENABLE                  false
#  define CONF_CLOCK_XOSC_EXTERNAL_CRYSTAL        SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC_EXTERNAL_FREQUENCY      12000000UL
#  define CONF_CLOCK_XOSC_STARTUP_TIME            SYSTEM_XOSC_STARTUP_32768
#  define CONF_CLOCK_XOSC_AUTO_GAIN_CONTROL       true
#  define CONF_CLOCK_XOSC_ON_DEMAND               true
#  define CONF_CLOCK_XOSC_RUN_IN_STANDBY          false

/* SYSTEM_CLOCK_SOURCE_XOSC32K configuration - External 32KHz crystal/clock oscillator */
#  define CONF_CLOCK_XOSC32K_ENABLE               false
#  define CONF_CLOCK_XOSC32K_EXTERNAL_CRYSTAL     SYSTEM_CLOCK_EXTERNAL_CRYSTAL
#  define CONF_CLOCK_XOSC32K_STARTUP_TIME         SYSTEM_XOSC32K_STARTUP_65536
#  define CONF_CLOCK_XOSC32K_AUTO_AMPLITUDE_CONTROL  false
#  define CONF_CLOCK_XOSC32K_ENABLE_1KHZ_OUPUT    false
#  define CONF_CLOCK_XOSC32K_ENABLE_32KHZ_OUTPUT  true
#  define CONF_CLOCK_XOSC32K_ON_DEMAND            true
#  define CONF_CLOCK_XOSC32K_RUN_IN_STANDBY       false

/* SYSTEM_CLOCK_SOURCE_OSC32K configuration - Internal 32KHz oscillator */
#  define CONF_CLOCK_OSC32K_ENABLE                false
#  define CONF_CLOCK_OSC32K_STARTUP_TIME          SYSTEM_OSC32K_STARTUP_130
#  define CONF_CLOCK_OSC32K_ENABLE_1KHZ_OUTPUT    true
```

```
#  define CONF_CLOCK_OSC32K_ENABLE_32KHZ_OUTPUT   true
#  define CONF_CLOCK_OSC32K_ON_DEMAND             true
#  define CONF_CLOCK_OSC32K_RUN_IN_STANDBY        false

/* SYSTEM_CLOCK_SOURCE_DFLL configuration - Digital Frequency Locked Loop */
#  define CONF_CLOCK_DFLL_ENABLE                  true
#  define CONF_CLOCK_DFLL_LOOP_MODE               SYSTEM_CLOCK_DFLL_LOOP_MODE_USB_RECOVERY
#  define CONF_CLOCK_DFLL_ON_DEMAND               true

/* DFLL open loop mode configuration */
#  define CONF_CLOCK_DFLL_COARSE_VALUE            (0x1f / 4)
#  define CONF_CLOCK_DFLL_FINE_VALUE              (0xff / 4)

/* DFLL closed loop mode configuration */
#  define CONF_CLOCK_DFLL_SOURCE_GCLK_GENERATOR   GCLK_GENERATOR_1
#  define CONF_CLOCK_DFLL_MULTIPLY_FACTOR         (48000000 / 32768)
#  define CONF_CLOCK_DFLL_QUICK_LOCK              true
#  define CONF_CLOCK_DFLL_TRACK_AFTER_FINE_LOCK   true
#  define CONF_CLOCK_DFLL_KEEP_LOCK_ON_WAKEUP     true
#  define CONF_CLOCK_DFLL_ENABLE_CHILL_CYCLE      true
#  define CONF_CLOCK_DFLL_MAX_COARSE_STEP_SIZE    (0x1f / 4)
#  define CONF_CLOCK_DFLL_MAX_FINE_STEP_SIZE      (0xff / 4)

/* SYSTEM_CLOCK_SOURCE_DPLL configuration - Digital Phase-Locked Loop */
#  define CONF_CLOCK_DPLL_ENABLE                  false
#  define CONF_CLOCK_DPLL_ON_DEMAND               true
#  define CONF_CLOCK_DPLL_RUN_IN_STANDBY          false
#  define CONF_CLOCK_DPLL_LOCK_BYPASS             false
#  define CONF_CLOCK_DPLL_WAKE_UP_FAST            false
#  define CONF_CLOCK_DPLL_LOW_POWER_ENABLE        false

#  define CONF_CLOCK_DPLL_LOCK_TIME               SYSTEM_CLOCK_SOURCE_DPLL_LOCK_TIME_NO_TIMEOUT
#  define CONF_CLOCK_DPLL_REFERENCE_CLOCK         SYSTEM_CLOCK_SOURCE_DPLL_REFERENCE_CLOCK_REF0
#  define CONF_CLOCK_DPLL_FILTER                  SYSTEM_CLOCK_SOURCE_DPLL_FILTER_DEFAULT

#  define CONF_CLOCK_DPLL_REFERENCE_FREQUENCY     32768
#  define CONF_CLOCK_DPLL_REFEREMCE_DIVIDER       1
#  define CONF_CLOCK_DPLL_OUTPUT_FREQUENCY        48000000

/* Set this to true to configure the GCLK when running clocks_init. If set to
 * false, none of the GCLK generators will be configured in clocks_init(). */
#  define CONF_CLOCK_CONFIGURE_GCLK               true

/* Configure GCLK generator 0 (Main Clock) */
#  define CONF_CLOCK_GCLK_0_ENABLE                true
#  define CONF_CLOCK_GCLK_0_RUN_IN_STANDBY        true
#  define CONF_CLOCK_GCLK_0_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_DFLL
#  define CONF_CLOCK_GCLK_0_PRESCALER             1
#  define CONF_CLOCK_GCLK_0_OUTPUT_ENABLE         false

/* Configure GCLK generator 1 */
#  define CONF_CLOCK_GCLK_1_ENABLE                false
#  define CONF_CLOCK_GCLK_1_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_1_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_XOSC32K
#  define CONF_CLOCK_GCLK_1_PRESCALER             1
#  define CONF_CLOCK_GCLK_1_OUTPUT_ENABLE         false

/* Configure GCLK generator 2 (RTC) */
#  define CONF_CLOCK_GCLK_2_ENABLE                false
#  define CONF_CLOCK_GCLK_2_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_2_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC32K
```

```
#  define CONF_CLOCK_GCLK_2_PRESCALER             32
#  define CONF_CLOCK_GCLK_2_OUTPUT_ENABLE         false

/* Configure GCLK generator 3 */
#  define CONF_CLOCK_GCLK_3_ENABLE                false
#  define CONF_CLOCK_GCLK_3_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_3_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_3_PRESCALER             1
#  define CONF_CLOCK_GCLK_3_OUTPUT_ENABLE         false

/* Configure GCLK generator 4 */
#  define CONF_CLOCK_GCLK_4_ENABLE                false
#  define CONF_CLOCK_GCLK_4_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_4_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_4_PRESCALER             1
#  define CONF_CLOCK_GCLK_4_OUTPUT_ENABLE         false

/* Configure GCLK generator 5 */
#  define CONF_CLOCK_GCLK_5_ENABLE                false
#  define CONF_CLOCK_GCLK_5_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_5_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_5_PRESCALER             1
#  define CONF_CLOCK_GCLK_5_OUTPUT_ENABLE         false

/* Configure GCLK generator 6 */
#  define CONF_CLOCK_GCLK_6_ENABLE                false
#  define CONF_CLOCK_GCLK_6_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_6_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_6_PRESCALER             1
#  define CONF_CLOCK_GCLK_6_OUTPUT_ENABLE         false

/* Configure GCLK generator 7 */
#  define CONF_CLOCK_GCLK_7_ENABLE                false
#  define CONF_CLOCK_GCLK_7_RUN_IN_STANDBY        false
#  define CONF_CLOCK_GCLK_7_CLOCK_SOURCE          SYSTEM_CLOCK_SOURCE_OSC8M
#  define CONF_CLOCK_GCLK_7_PRESCALER             1
#  define CONF_CLOCK_GCLK_7_OUTPUT_ENABLE         false

#endif /* CONF_CLOCKS_H_INCLUDED */
```

12.3.4 conf_board.h

12.3.4.1 SAM3X, and SAM3A Devices (UOTGHS: USB OTG High Speed)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// USB pins are used
#define CONF_BOARD_USB_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

12.3.4.2 SAM4L Device (USBC)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

// Auto-initialize USART GPIOs when board_init() is called
```

```
//#define CONF_BOARD_COM_PORT

// Enable USB interface (USB)
#define CONF_BOARD_USB_PORT

#endif /* CONF_BOARD_H_INCLUDED */
```

12.3.4.3 SAMD21 Devices (USB)

```
#ifndef CONF_BOARD_H_INCLUDED
#define CONF_BOARD_H_INCLUDED

/* Enable USB VBUS detect */
#define CONF_BOARD_USB_VBUS_DETECT

#endif /* CONF_BOARD_H_INCLUDED */
```

Index

E

F

M

Macro Definitions

P

Public Variable Definitions

S

Structure Definitions

T

Type Definitions

Document Revision History

Doc. Rev.	Date	Comments
42336A	12/2014	Initial release.

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**